Tomus 8. Fasciculus 4.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

Szeged, 1988
Curat: Universitas Szegediensis de Attila József nominata

8. kötet 4. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK
KÖZPONTI PUBLIKÁCIÓS FÓRUMA

Szeged, 1988
A Szegedi József Attila Tudományegyetem gondozásában

Tomus 8.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

Szeged, 1988
Curat: Universitas Szegediensis de Attila József nominata

INDEX

Tomus 8.

88-1052 — Szegedi Nyomda — F. v.: Surányi Tibor igazgató

On homomorphic simulation of automata by α_0-products

P. DÖMÖSI and Z. ÉSIK

1. Introduction

The concept of the α_0-product of automata is equivalent to the cascade composition or loop-free product (see [1, 7]). In an α_0-product, the feedback functions admit only strict letter-to-letter replacement as opposed to the generalized α_0-product where input words may correspond to input letters. Thus the generalized α_0-product is closely related to the wreath product of transformation semigroups and/or monoids, see [1, 4]. The α_0-product and the above generalization are usually studied in conjunction with homomorphic realization or homomorphic simulation. The difference between the concepts of homomorphic realization and homomorphic simulation is similar to the difference between the α_0-product and the generalized α_0-product: for simulation the action of an input letter is related to the action of an input word rather than to the action of an input letter. It is a matter of fact that the homomorphic realization is equivalent to the homomorphic simulation with respect to the generalized α_0-product. In the present paper we study homomorphic simulations of automata by α_0-products. We give a sufficient condition on a class $\mathscr{K}$ of automata ensuring that an automaton be homomorphically simulated by a generalized α_0-product over $\mathscr{K}$ if and only if it is homomorphically simulated by an α_0-product of automata from $\mathscr{K}$. As an application it is shown that a class $\mathscr{K}$ is complete with respect to the homomorphic simulation by the generalized α_0-product if and only if it is complete with respect to the homomorphic simulation by the α_0-product, as far as nonempty words are considered.

2. Preliminaries

For a finite nonempty set X we let X^* denote the free monoid of all *words* over X and write X^+ for the free semigroup $X^*-\{\lambda\}$, where λ is the *empty word*. We set $X^\lambda=X\cup\{\lambda\}$. The *length* of a word $u\in X^*$ is denoted $|u|$. If $u=x_1\dots x_n$ with the x's in X, then for each $i\in[n]=\{1,\dots,n\}$ we define $u(i)=x_i$ and $u[i]=x_1\dots x_{i-1}$.

An *automaton* is a triple $\mathbf{A}=(A, X, \delta)$ with finite nonempty set A *(state set)*, X *(input letters)* and *transition* δ: $A\times X\to A$ that extends to a mapping $A\times X^*\to A$ as usual. If $u\in X^*$ we write $u^{\mathbf{A}}$ for the transformation $A\to A$ given by $au^{\mathbf{A}}=\delta(a, u)$,

$a \in A$. The *characteristic monoid (semigroup)* $S_1(\mathbf{A})$, $(S(\mathbf{A}))$ of $\mathbf{A}$ consists of all the transformations $u^{\mathbf{A}}$ with $u \in X^*$ $(u \in X^+)$.

Let $\mathbf{A}=(A, X, \delta)$ be an automaton. We define $\mathbf{A}^*=(A, S_1(\mathbf{A}), \delta^*)$ and $\mathbf{A}^+=(A, S(\mathbf{A}), \delta^+)$ to be the automata with $\delta^*(a, s)=as$ and $\delta^+(a, t)=at$, for all $a \in A$, $s \in S_1(\mathbf{A})$ and $t \in S(\mathbf{A})$. Likewise we put $\mathbf{A}^\lambda=(A, \{u^{\mathbf{A}} | u \in X^\lambda\}, \delta^\lambda)$ with $\delta^\lambda(a, u^{\mathbf{A}})=au^{\mathbf{A}}$. The automata $\mathbf{A}^*$ and $\mathbf{A}^+$ thus correspond to the transformation monoid and the transformation semigroup of $\mathbf{A}$, see [3].

Given a family of automata $\mathbf{A}_i=(A_i, X_i, \delta_i)$ $(i \in [n], n \geqq 0)$ and a finite non-empty set X together with *feedback functions*

$$\varphi_i: A_1 \times \ldots \times A_{i-1} \times X \to X_i,$$

the α_0-*product* (cf. [8])

$$\mathbf{A}_1 \times \ldots \times \mathbf{A}_n(X, \varphi)$$

is defined to be the automaton $\mathbf{A}=(A, X, \delta)$ with

$$A = A_1 \times \ldots \times A_n$$

and

$$\delta((a_1, \ldots, a_n). x) = (\delta_1(a_1, x_1), \ldots, \delta_n(a_n, x_n)),$$

$$x_i = \varphi_i(a_1, \ldots, a_{i-1}, x) \quad (i \in [n]),$$

for all $(a_1, \ldots, a_n) \in A$ and $x \in X$. The α_0-product is equivalent to the cascade composition or the loop-free product (cf. [1, 7]).

We let $\mathbf{H}$, $\mathbf{S}$ and $\mathbf{P}_{\alpha_0}$ denote the operator corresponding to the formation of homomorphic images, subautomata and α_0-products, resp. Thus, if $\mathscr{K}$ is a class of automata, then $\mathbf{P}_{\alpha_0}(\mathscr{K})$ is the class of all α_0-products of automata from $\mathscr{K}$. Further, we let $\mathbf{P}_{1\alpha_0}(\mathscr{K})$ be the class

$$\{\mathbf{A}(X, \varphi) | \mathbf{A} \in \mathscr{K}, \mathbf{A}(X, \varphi) \text{ is an } \alpha_0\text{-product}\}$$

and define $\mathscr{K}^* = \cup(\mathbf{P}_{1\alpha_0}(\mathbf{A}^*) | \mathbf{A} \in \mathscr{K})$, $\mathscr{K}^+ = \cup(\mathbf{P}_{1\alpha_0}(\mathbf{A}^+) | \mathbf{A} \in \mathscr{K})$ and $\mathscr{K}^\lambda = \cup(\mathbf{P}_{1\alpha_0}(\mathbf{A}^\lambda) | \mathbf{A} \in \mathscr{K})$.

If $\mathbf{O}$ is one of the operators $\mathbf{S}$ and $\mathbf{P}_{\alpha_0}$, then by $\mathbf{O}^*(\mathscr{K})$ $(\mathbf{O}^+(\mathscr{K}), \mathbf{O}^\lambda(\mathscr{K}))$ we denote the class $\mathbf{O}(\mathscr{K}^*)$ $(\mathbf{O}(\mathscr{K}^+), \mathbf{O}(\mathscr{K}^\lambda))$. We have $\mathbf{P}^*_{\alpha_0}(\mathscr{K}) = \mathbf{P}_{\alpha_0}(\{\mathbf{A}^* | \mathbf{A} \in \mathscr{K}\})$, $\mathbf{P}^+_{\alpha_0}(\mathscr{K}) = \mathbf{P}_{\alpha_0}(\{\mathbf{A}^+ | \mathbf{A} \in \mathscr{K}\})$ and $\mathbf{P}^\lambda_{\alpha_0}(\mathscr{K}) = \mathbf{P}_{\alpha_0}(\{\mathbf{A}^\lambda | \mathbf{A} \in \mathscr{K}\})$. Moreover, $\mathbf{A} \in \mathbf{HS}^*(\{\mathbf{B}\})$ for automata $\mathbf{A}=(A, X, \delta)$ and $\mathbf{B}=(B, Y, \delta')$ if and only if, there exist a set $B' \subseteq B$, an onto mapping $h: B' \to A$ and a mapping $\varphi: X \to Y^*$ such that $\delta(h(b), x) = h(\delta'(b, \varphi(x)))$ for all $b \in B'$ and $x \in X$. It is understood that $\delta'(b, \varphi(x)) \in B'$. Similar fact is true for the combined operators $\mathbf{HS}^+$ and $\mathbf{HS}^\lambda$. In [6] the relation $\mathbf{A} \in \mathbf{HS}^*(\{\mathbf{B}\})$ is expressed by saying that $\mathbf{A}$ is *homomorphically simulated* by $\mathbf{B}$. We also note that the operators $\mathbf{HS}^*$ and $\mathbf{HS}^+$ correspond to the covering relation (or division) of transformation monoids and/or transformation semigroups, see [4].

In the sequel we shall also make use of another view of the operators $\mathbf{P}^*_{\alpha_0}$, $\mathbf{P}^+_{\alpha_0}$ and $\mathbf{P}_{\alpha_0}$. Define the concept of the α_0^*-*product* (α_0^+-*product*, α_0^λ-*product*) in exact analogue with the α_0-product except for the fact that each feedback function φ assumes values in X_i^* (X_i^+, X_i^λ). In this setting $\mathbf{P}^*_{\alpha_0}$ ($\mathbf{P}^+_{\alpha_0}$, $\mathbf{P}^\lambda_{\alpha_0}$) becomes the operator of forming α_0^*-products (α_0^+-products, α_0^λ-products). It is apparent that the generalized α_0-products, i.e. the α_0^*-product and the α_0^+-product, are closely related to the wreath product of transformation semigroups, cf. [4].

The above defined operators and the combined ones, e.g. $\mathbf{HS^*P}_{\alpha_0}$, satisfy a number of simple closure properties that we shall use implicitly. In this paper the emphasis will be on the combinations $\mathbf{HS^*P}^*_{\alpha_0}$ vs $\mathbf{HS^*P}_{\alpha_0}$, and also on $\mathbf{HS^+P}^+_{\alpha_0}$ and $\mathbf{HS^+P}_{\alpha_0}$.

Also the operators $\mathbf{HSP}^*_{\alpha_0}$ and $\mathbf{HSP}^+_{\alpha_0}$ could be of interest. These are however discarded due to the following simple fact, see also [7].

Proposition 2.1. For every class $\mathscr{K}$ and modifier $m\in\{*, +, \lambda\}$ it holds that $\mathbf{S}^m\mathbf{P}_{\alpha_0}(\mathscr{K})\subseteq\mathbf{S}^m\mathbf{P}^m_{\alpha_0}(\mathscr{K})=\mathbf{SP}^m_{\alpha_0}(\mathscr{K})$.

The inclusion $\mathbf{HS^*P}_{\alpha_0}(\mathscr{K})\subseteq\mathbf{HS^*P}^+_{\alpha_0}(\mathscr{K})$, just as $\mathbf{HS^+P}_{\alpha_0}(\mathscr{K})\subseteq\mathbf{HS^+P}^+_{\alpha_0}(\mathscr{K})$, cannot usually be turned to equality. E.g. if $\mathscr{K}$ consists of a single counter with prime length $p>1$, i.e. $\mathscr{K}=\{\mathbf{C}\}$ with $\mathbf{C}=([p], \{x\}, \delta)$, $\delta(i, x)\equiv i+1 \bmod p$, then $\mathbf{HS^*P}_{\alpha_0}(\mathscr{K})=\mathbf{HS^+P}_{\alpha_0}(\mathscr{K})$ consist of commutative automata with very simple structure. On the other hand, $\mathbf{HS^+P}^*_{\alpha_0}(\mathscr{K})=\mathbf{HS^+P}^+_{\alpha_0}(\mathscr{K})$ is the class of all automata that could be called p-automata: i.e. permutation automata whose characteristic monoid is a p-group. The latter observation follows from the Krohn—Rhodes Decomposition Theorem, see below. In the next section there is given a sufficient condition ensuring $\mathbf{HS^*P}^*_{\alpha_0}(\mathscr{K})=\mathbf{HS^*P}_{\alpha_0}(\mathscr{K})$. In fact the condition will quarantee that $\mathbf{HS^*P}^*_{\alpha_0}(\mathscr{K})=\mathbf{HS^*P}_{\alpha_0}(\mathscr{K})=\mathbf{HS^+P}^+_{\alpha_0}(\mathscr{K})=\mathbf{HS^+P}_{\alpha_0}(\mathscr{K})$.

Some more terminology. By a semigroup we always mean a finite semigroup. We put $S|T$, i.e., S divides T, for semigroups S and T, if and only if S is a homomorphic image of a subsemigroup of T. If S is a monoid (group), it is equivalent to saying that S is a homomorphic image of a submonoid (subgroup) of T, see [1, 4]. (When talking about a submonoid M of a semigroup S which is a monoid, M is not required to contain the identity of S.) The following fact is known, see [4] and also [7] for the group case.

Lemma 2.2. Let $\mathbf{A}=(A, X, \delta)$ be an automaton and M a submonoid of $S(\mathbf{A})$ or $S_1(\mathbf{A})$. There exists a nonempty set $B\subseteq A$ with the following properties:

(i) The elements of M map B into itself.

(ii) The restriction of the identity of M to B is the identical mapping $B\to B$.

(iii) If m_1 and m_2 are distinct elements of M then $m_1(b)\neq m_2(b)$ for at least one $b\in B$.

To end this section we mention one more useful fact whose proof is omitted. For similar, in fact stronger statements, see [4]. A trivial automaton is an automaton with a single state.

Lemma 2.3. If $S_1(\mathbf{A})|S_1(\mathbf{B})$ for two automata $\mathbf{A}$, $\mathbf{B}$ and either $\mathbf{B}$ is nontrivial or $\mathbf{A}$ is trivial, then $\mathbf{A}\in\mathbf{HSP}^*_{\alpha_0}(\{\mathbf{B}\})$.

3. The results

We start with an auxiliary definition. Let $\mathbf{A}=\mathbf{A}_1\times\ldots\times\mathbf{A}_n(X, \varphi)$ $(n\geq 1)$ be an α_0^+-product with components $\mathbf{A}_i=(A_i, X_i, \delta_i)$ and let $\mathbf{B}=(B, X, \delta)$ be a subautomaton of $\mathbf{A}$. For an integer $i\in[n]$ the *useful* states of $\mathbf{A}_i$ (with respect to $\mathbf{B}$) are those states in A_i which occur in the place of the i-th component of the elements of B.

Lemma 3.1. Let $\mathbf{A}$ and $\mathbf{B}$ be automata as above. Suppose that for each $x\in X$ an integer $k_x\geqq 1$ is given with

$$|\varphi_i(a_1, \ldots, a_{i-1}, x)| = k_x,$$

for all $i\in[n]$ and $(a_1, \ldots, a_{i-1})\in A_1\times\ldots\times A_{i-1}$. Assume further that for each i and x there is a word $p_i^x\in X_i^+$ with $|p_i^x|=k_x$ and $\delta_i(a_i, p_i^x)=a_i$ whenever $a_i\in A_i$ is useful. Then $\mathbf{B}$ is isomorphic to an automaton in $\mathbf{S}^+(\{\mathbf{A}'\})$ for an α_0-product $\mathbf{A}'=\mathbf{A}_1\times\ldots\times\mathbf{A}_n(Y, \psi)$.

Proof. For every $x\in X$ let Y_x be a new set of $m_x=nk_x$ input letters, say

$$Y_x = \{y_j^x | j\in[m_x]\}.$$

Set $Y=\bigcup(Y_x|x\in X)$. To define the feedback function ψ_i, $i\in[n]$, let $a_1\in A_1, \ldots, a_{i-1}\in A_{i-1}$ be fixed states and $y_j^x\in Y_x$. Let $t\in[k_x]$ be that integer with $j\equiv t \bmod k_x$. If there is an $s\in[i-1]$ such that a_s is not of the form $\delta_s(b_s, p_s^x[t])$ for some useful state $b_s\in A_s$, then $\psi_i(a_1, \ldots, a_{i-1}, y_j^x)$ is any letter in X_i. Otherwise there are uniquely determined useful states $b_1\in A_1, \ldots, b_{i-1}\in A_{i-1}$ with $\delta_s(b_s, p_x^s[t])=a_s$, $s\in[i-1]$. If $j\leqq(i-1)k_x$ or $j>ik_x$ then we define

$$\psi_i(a_1, \ldots, a_{i-1}, y_j^x) = p_i^x(t).$$

Finally, if $(i-1)k_x<j\leqq ik_x$, we put

$$\psi_i(a_1, \ldots, a_{i-1}, y_j^x) = q(t),$$

where

$$q = \varphi_i(b_1, \ldots, b_{i-1}, x).$$

This ends the definition of the α_0-product $\mathbf{A}'$.

Let $x\in X$ be any letter and define

$$u^x = u_n^x \ldots u_1^x, u_j^x = y^x_{(j-k)k_x+1}\cdots y^x_{jk_x},$$

$(j\in[n])$. Denote by δ' the transition of $\mathbf{A}'$. To see that $\mathbf{B}$ is in $\mathbf{S}^+(\{\mathbf{A}'\})$, it suffices to show that for any $b=(b_1, \ldots, b_n)\in B$ and $x\in X$ we have $\delta'(b, u^x)=\delta(b, x)$. This is however obvious, for if $\delta(b, x)=c=(c_1, \ldots, c_n)$, then for each $i\in[n]$ we can compute as follows:

$$\begin{aligned}\delta'((b_1, \ldots, b_{i-1}, b_i, c_{i+1}, \ldots, c_n), u_i^x) &= \\ = (\delta_1(b_1, p_1^x), \ldots, \delta_{i-1}(b_{i-1}, p_{i-1}^x)), & \\ \delta_i(b_i, \varphi_i(b_1, \ldots, b_{i-1}, x)), & \\ \delta_{i+1}(c_{i+1}, p_{i+1}^x), \ldots, \delta_n(c_n, p_n^x)) &= \\ = (b_1, \ldots, b_{i-1}, c_i, c_{i+1}, \ldots, c_n). &\end{aligned}$$

A straightforward induction argument completes the proof.

Recall that a permutation automaton $\mathbf{A}=(A, X, \delta)$ is an automaton such that δ_x is a permutation of the state set for each $x\in X$. Equivalently, $\mathbf{A}$ is a permutation automaton if and only if $S_1(\mathbf{A})$ is a group. Note that $S_1(\mathbf{A})=S(\mathbf{A})$ for a permutation automaton.

Remark. If the automata $\mathbf{A}_i$ of the previous lemma were permutation automata, then a much simpler argument could be applied. In fact we could define

$$Y_x = \{y_j^x | j \in [k_x]\}, \quad Y = \cup (Y_x | x \in X),$$

and then

$$\psi_i(a_1, \ldots, a_{i-1}, y_j^x) = (\varphi_i(b_1, \ldots, b_{i-1}, x))(j),$$

where the states b_s, $s \in [n]$, are successively determined by the condition

$$\delta_s(b_s, (\varphi_s(b_1, \ldots, b_{s-1}, x))[j]) = a_s.$$

For a more general form of the following definition see [4]. Let M be a monoid and $\mathbf{A}=(A, X, \delta)$ an automaton. We write $M \| S(\mathbf{A})$ $(M \| S_1(\mathbf{A}))$ if and only if there exists a submonoid M' of $S(\mathbf{A})$ $(S_1(\mathbf{A}))$ which can be mapped homomorphically onto M and such that $M' \subseteq \{u^{\mathbf{A}} | u \in X^*, |u|=n\}$ for an integer $n>0$ $(n \geqq 0)$. Notice that $M \| S(\mathbf{A})$ if and only if $M \| S_1(\mathbf{A})$, for if $M \| S_1(\mathbf{A})$ with $n=0$ then M' is trivial and so is M.

Theorem 3.2. Let $\mathscr{K}_1$ and $\mathscr{K}_2$ be two classes of automata. Assume that $\mathscr{K}_1$ contains an automaton $\mathbf{A}_0$ such that $S_1(\mathbf{A}_0)$ is a nontrivial monoid. Assume further that for every $\mathbf{A} \in \mathscr{K}_1$ there is $\mathbf{B} \in \mathscr{K}_2$ with $S_1(\mathbf{A}) \| S(\mathbf{B})$. Then $\mathbf{HS}^* \mathbf{P}^*_{\alpha_0}(\mathscr{K}_1) \subseteq \subseteq \mathbf{HS}^+ \mathbf{P}^+_{\alpha_0}(\mathscr{K}_2)$.

Proof. First note that $\mathbf{HS}^* \mathbf{P}^*_{\alpha_0}(\mathscr{K}_1) = \mathbf{HS}^* \mathbf{P}^*_{\alpha_0}(\mathscr{K}_1 - \mathscr{K}_0)$, where $\mathscr{K}_0$ is the class of all automata with trivial characteristic monoid. (The class $\mathscr{K}_0$ can also be called the class of discrete automata, for an automaton belongs to $\mathscr{K}_0$ if and only if each input letter induces the identical state transformation.) Thus it suffices to prove that $\mathbf{HS}^* \mathbf{P}^*_{\alpha_0}(\mathscr{K}_1 - \mathscr{K}_0) \subseteq \mathbf{HS}^+ \mathbf{P}_{\alpha_0}(\mathscr{K}_2)$; or even, by Proposition 2.1, it is enough to show the inclusion $\mathbf{P}^*_{\alpha_0}(\mathscr{K}_1 - \mathscr{K}_0) \subseteq \mathbf{HS}^+ \mathbf{P}_{\alpha_0}(\mathscr{K}_2)$.

Let $\mathbf{A} = \mathbf{A}_1 \times \ldots \times \mathbf{A}_n(X, \varphi)$ be any α_0^*-product with components $\mathbf{A}_i \in \mathscr{K}_1 - \mathscr{K}_0$. If $n=0$ then $\mathbf{A}$ is trivial, so that $\mathbf{A} \in \mathbf{HS}^+ \mathbf{P}_{\alpha_0}(\mathscr{K}_2)$. Assume $n>0$. For every $i \in [n]$ there are an automaton $\mathbf{B}_i=(B_i, X_i, \delta^i) \in \mathscr{K}_2$, a submonoid M_i of $S(\mathbf{B}_i)$ and an integer $k_i>0$ such that $M_i \subseteq \{u^{\mathbf{B}_i} | u \in X_i^+, |u|=k_i\}$ and $S_1(\mathbf{A}_i)$ is a homomorphic image of M_i. Let k be the l.c.m. of the numbers k_i. If $u_0^{\mathbf{B}_i}$ is the identity of M_i and $|u_0|=k_i$, then for any $u^{\mathbf{B}_i} \in M_i$ with $|u|=k_i$ we have $u^{\mathbf{B}_i} = w^{\mathbf{B}_i}$ where $w = u u_0^{k/k_i - 1}$. It follows that $M_i \subseteq \{u^{\mathbf{B}_i} | u \in X_i^+, |u|=k\}$.

Let $i \in [n]$ be a fixed integer. Since M_i is a submonoid of $S(\mathbf{B}_i)$, there is a (nonempty) set $B_i' \subseteq B_i$ as in Lemma 2.2. Define the automaton $\mathbf{B}_i'=(B_i', M_i, \delta_i)$ by $\delta_i(b, m)=m(b)$, for all $b \in B_i'$ and $m \in M$. M_i is isomorphic to $S_1(B_i')$ and every transformation in $S_1(B_i')$ is induced by a letter in M_i. Since $S_1(\mathbf{A}_i)$ is a homomorphic image of $S_1(B_i')$ and $S_1(A_i)$ is nontrivial, from Lemma 2.3 we obtain $\mathbf{A}_i \in \mathbf{HSP}^*_{\alpha_0}(\{B_i'\})$.

We have seen that $\mathbf{A}_i \in \mathbf{HSP}^*_{\alpha_0}(\{\mathbf{B}_i'\})$ for all i. Consequently also $\mathbf{A} \in \in \mathbf{HSP}^*_{\alpha_0}(\{\mathbf{B}_1', \ldots, \mathbf{B}_n'\})$, and since the members of each $S_1(\mathbf{B}_i')$ are induced by input letters, $\mathbf{A} \in \mathbf{HSP}_{\alpha_0}(\{\mathbf{B}_1', \ldots, \mathbf{B}_n'\})$. Let $\mathbf{B}' = \mathbf{B}_{i_1}' \times \ldots \times \mathbf{B}_{i_t}'(X, \varphi')$ be an α_0-product of the automata $\mathbf{B}_1', \ldots, \mathbf{B}_n'$ containing a subautomaton which can be mapped homomorphically onto $\mathbf{A}$. We define an α_0^+-product $\mathbf{B} = \mathbf{B}_{i_1} \times \ldots \times \mathbf{B}_{i_t}(X, \psi)$ as follows. For each $j \in [t]$, let $u_j \in X_{i_j}^+$ be a fixed word with $|u_j|=k$, and to each $(b_1, \ldots, b_{j-1}) \in \in B_{i_1}' \times \ldots \times B_{i_{j-1}}'$ and $x \in X$ let us correspond a word $u = u(b_1, \ldots, b_{j-1}, x) \in X_{i_j}^+$ with $|u|=k$ and $u^{\mathbf{B}_{i_j}} = \varphi_j'(b_1, \ldots, b_{j-1}, x)$. Then for all $j \in [t]$, $(b_1, \ldots, b_{j-1}) \in$

$\in B_{i_1}\times\ldots\times B_{i_{j-1}}$ and $x\in X$ let

$$\psi_j(b_1, \ldots, b_{j-1}, x) = \begin{cases} u(b_1, \ldots, b_{j-1}, x) & \text{if } (b_1, \ldots, b_{j-1})\in B'_{i_1}\times\ldots B'_{i_{j-1}}, \\ u_j & \text{otherwise.} \end{cases}$$

It is easy to see that $\mathbf{B}$ contains an isomorphic copy of $\mathbf{B}'$, in fact $\mathbf{B}'$ is a subautomaton of $\mathbf{B}$. The α_0^+-product $\mathbf{B}$ and the subautomaton $\mathbf{B}'$ satisfies the assumptions of Lemma 3.1, therefore $\mathbf{B}'\in\mathbf{S}^+\mathbf{P}_{\alpha_0}(\{\mathbf{B}_1, \ldots, \mathbf{B}_n\})\subseteq\mathbf{S}^+\mathbf{P}_{\alpha_0}(\mathscr{K}_2)$. Since $\mathbf{A}\in\mathbf{HS}(\{\mathbf{B}'\})$ it follows that $\mathbf{A}\in\mathbf{HS}^+\mathbf{P}_{\alpha_0}(\mathscr{K}_2)$. The proof is complete.

Notice that also $\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K}_1)\subseteq\mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K}_2)$ and $\mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K}_1)\subseteq\mathbf{HS}^+\mathbf{P}_{\alpha_0}(\mathscr{K}_2)$.

It should be noted that if $\mathscr{K}_1$ consists of discrete automata one of which is nontrivial, then $\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K}_1)=\mathscr{K}_0$, the class of all discrete automata. Moreover, $\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K}_1)\subseteq\mathbf{HS}^+\mathbf{P}_{\alpha_0}(\mathscr{K}_2)$ if and only if $\mathscr{K}_2$ contains an automaton $\mathbf{A}$ which is not definite, i.e., which has two distinct states a_1, a_2 and a nonempty input word u with $a_iu^{\mathbf{A}}=a_i$, $i=1, 2$.

Next we give a reformulation of Theorem 3.2 and discuss some consequences. For a monoid M, define $\mathrm{Aut}(M)=(M, M, \delta)$ with $\delta(m_1, m_2)=m_1m_2$. If $\mathscr{M}$ is a class of monoids, set $\mathrm{Aut}(\mathscr{M})=\{\mathrm{Aut}(M)|M\in\mathscr{M}\}$.

Corollary 3.3. Let $\mathscr{M}$ be a class of monoids and $\mathscr{K}$ a class of automata. Suppose that for each $M\in\mathscr{M}$ there is az automaton $\mathbf{A}\in\mathscr{K}$ with $M\|S(\mathbf{A})$. Then $\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathrm{Aut}(\mathscr{M}))=\mathbf{HSP}_{\alpha_0}(\mathrm{Aut}(\mathscr{M}))\subseteq\mathbf{HS}^+\mathbf{P}_{\alpha_0}(\mathscr{K})$.

Corollary 3.4. Let $\mathscr{K}_1$ and $\mathscr{K}_2$ be two classes of automata such that for each $\mathbf{A}\in\mathscr{K}_1$ there is an automaton $\mathbf{B}\in\mathscr{K}_2$ with $S_1(\mathbf{A})|S_1(\mathbf{B})$. Suppose further that either $\mathscr{K}_1$ consists of trivial automata or $\mathscr{K}_2$ contains a nontrivial automaton. Then $\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K}_1)\subseteq\mathbf{HS}^+\mathbf{P}^\lambda_{\alpha_0}(\mathscr{K}_2)$.

Proof. If $\mathscr{K}_1$ consists of discrete automata then the result is obvious. Otherwise there is an automaton $\mathbf{A}_0\in\mathscr{K}_1$ such that $S_1(\mathbf{A}_0)$ is nontrivial. If $S_1(\mathbf{A})|S_1(\mathbf{B})$ then $S_1(\mathbf{A})\|S(\mathbf{B}^\lambda)$. Thus the inclusion $\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K}_1)\subseteq\mathbf{HS}^+\mathbf{P}^\lambda_{\alpha_0}(\mathscr{K}_2)$ is obtained by applying Theorem 3.2 for $\mathscr{K}_1$ and $\mathscr{K}_2^\lambda$.

Corollary 3.5. Let $\mathscr{K}$ be any class of automata. If for every $\mathbf{A}\in\mathscr{K}$ there exists $\mathbf{B}\in\mathscr{K}$ with $S_1(\mathbf{A})\|S(\mathbf{B})$ then $\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K})=\mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})=\mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K})=\mathbf{HS}^+\mathbf{P}_{\alpha_0}(\mathscr{K})$. Moreover, $\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K})=\mathbf{HS}^+\mathbf{P}^\lambda_{\alpha_0}(\mathscr{K})$ holds universally.

The Krohn—Rhodes Decomposition Theorem (cf. [1, 4, 7]) is a basis for studying the α_0-product. Below we give one possible formalization in terms of the operators $\mathbf{H}$, $\mathbf{S}^+$, $\mathbf{S}^*$, $\mathbf{P}^+_{\sigma_0}$ and $\mathbf{P}^*_{\alpha_0}$. Following [1], by U_3 we denote the three-element monoid with two right zeros. An irreducible semigroup is a semigroup S such that for every nonempty class $\mathscr{K}$, if $S|S_1(\mathbf{A})$ for some $\mathbf{A}\in\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K})$ then there is an automaton $\mathbf{B}\in\mathscr{K}$ with $S|S_1(\mathbf{B})$. Equivalently this means that for every nonempty class $\mathscr{K}$, if $S|S(\mathbf{A})$ for an automaton $\mathbf{A}\in\mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K})$ or $\mathbf{A}\in\mathbf{HSP}_{\alpha_0}(\mathscr{K})$ then $S|S(\mathbf{B})$ for some $\mathbf{B}\in\mathscr{K}$. Notice that for a group G the conditions $G|S_1(\mathbf{A})$ and $G|S(\mathbf{A})$ are equivalent.

Theorem 3.6. *Krohn—Rhodes Decomposition Theorem.*

(1) For every group G let $\mathbf{A}_G$ be any automaton with $G|S(\mathbf{A}_G)$ and let $\mathbf{A}_0$ be an automaton with $U_3|S_1(\mathbf{A}_0)$ $(U_3|S(\mathbf{A}_0))$. Given an automaton $\mathbf{A}$, define $\mathscr{K}=\{\mathbf{A}_G|G$ is a simple group with $G|S(\mathbf{A})\}$. Then $\mathbf{A}\in\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K}\cup\{\mathbf{A}_0\})$

$(\mathbf{A}\in\mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K}\cup\{\mathbf{A}_0\}))$. If $\mathbf{A}$ is a permutation automaton and $S_1(\mathbf{A})$ is nontrivial, then $\mathbf{A}\in\mathbf{HS}^+\mathbf{P}^*_{\alpha_0}(\mathscr{K})$.

(2) A semigroup S is irreducible if and only if S is a simple group or $S|U_3$.

The monoids M with $M|U_3$, $M\neq U_3$, are the trivial monoid and the two-element monoid U_2 with a right zero. Let $\mathscr{G}$ be a nonempty class of simple groups closed under division, i.e. such that $G\in\mathscr{G}$ and $H|G$ implies $H\in\mathscr{G}$ for every simple group H. We define:

$$\mathscr{K}_3(\mathscr{G}) = \mathbf{HSP}_{\alpha_0}(\mathrm{Aut}(\mathscr{G}\cup\{U_3\})),$$

$$\mathscr{K}_2(\mathscr{G}) = \mathbf{HSP}_{\alpha_0}(\mathrm{Aut}(\mathscr{G}\cup\{U_2\})),$$

$$\mathscr{K}_0(\mathscr{G}) = \mathbf{HSP}_{\alpha_0}(\mathrm{Aut}(\mathscr{G})).$$

Note that $\mathscr{K}_3(\mathscr{G})=\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathrm{Aut}(\mathscr{G}\cup\{U_3\}))$ and similarly for $\mathscr{K}_2(\mathscr{G})$ and $\mathscr{K}_0(\mathscr{G})$. The avoid trivial situations, when writing $\mathscr{K}_0(\mathscr{G})$, we shall always assume that $\mathscr{G}$ contains a nontrivial group. As a direct consequence of the Krohn—Rhodes Decomposition Theorem we have:

Corollary 3.7.

(i) $\mathscr{K}_3(\mathscr{G})\subseteq\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K})$ $(\mathscr{K}_3(\mathscr{G})\subseteq\mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K}))$ if and only if the following hold:
  (i_1) For every $G\in\mathscr{G}$ there is $\mathbf{A}\in\mathscr{K}$ with $G|S(\mathbf{A})$.
  (i_2) There is an automaton $\mathbf{A}\in\mathscr{K}$ with $U_3|S_1(\mathbf{A})$ $(U_3|S(\mathbf{A}))$.

(ii) $\mathscr{K}_2(\mathscr{G})\subseteq\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K})$ $(\mathscr{K}_2(\mathscr{G})\subseteq\mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K}))$ if and only if (i_1) and (ii_1) hold:
  (ii_1) There is $\mathbf{A}\in\mathscr{K}$ with $U_2|S_1(\mathbf{A})$ $(U_2|S(\mathbf{A}))$.

(iii) $\mathscr{K}_0(\mathscr{G})\subseteq\mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K})$ $(\mathscr{K}_0(\mathscr{G})\subseteq\mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K}))$ if and only if (i_1) holds.

We note that $U_2|S_1(\mathbf{A})$ for an automaton $\mathbf{A}$ if and only if $\mathbf{A}$ is not a permutation automaton. In order to establish similar results for the operators $\mathbf{HS}^*\mathbf{P}_{\alpha_0}$ and $\mathbf{HS}^+\mathbf{P}_{\alpha_0}$, we need the following facts. Proposition 3.8 derives from a strong result in [2], for a direct proof see also [6].

Proposition 3.8. Let G be any group and $\mathbf{A}$ an automaton. If $G|S(\mathbf{A})$ then $G'\|S(\mathbf{A})$, where G' denotes the commutator group of G.

Corollary 3.9. Let G be a nonabelian simple group and $\mathbf{A}$ an automaton. If $G|S(\mathbf{A})$ then $G\|S(\mathbf{A})$.

Proposition 3.10. If for $i=2, 3$ we have $U_i|S(\mathbf{A})$ then $U_i\|S(\mathbf{A})$.

Proposition 3.11. Let G be a nontrivial simple group. If $G\|S(\mathbf{A})$ for an automaton $\mathbf{A}\in\mathbf{HSP}\alpha_0(\mathscr{K})$, where $\mathscr{K}$ is any class of automata, then $G\|S(\mathbf{B})$ for some $\mathbf{B}\in\mathscr{K}$.

The proof of Proposition 3.10 is trivial. Proposition 3.11 is from [5]. In the rest of the paper $\mathscr{G}$ denotes a fixed class of simple groups closed under division. Recall that when dealing with $\mathscr{K}_0(\mathscr{G})$ it is assumed that $\mathscr{G}$ contains a nontrivial group.

Theorem 3.12. Let $\mathscr{K}$ be a class of automata.
(i) $\mathscr{K}_3(\mathscr{G}) \subseteq \mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})$ if and only if (i_1)—(i_3) hold:
  (i_1) For every nonabelian $G \in \mathscr{G}$ there is $\mathbf{A} \in \mathscr{K}$ with $G|S(\mathbf{A})$.
  (i_2) For every abelian $G \in \mathscr{G}$ there is $\mathbf{A} \in \mathscr{K}$ with $G \| S(\mathbf{A})$.
  (i_3) There is an automaton $\mathbf{A} \in \mathscr{K}$ with $U_3|S(\mathbf{A})$.
(ii) $\mathscr{K}_2(\mathscr{G}) \subseteq \mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})$ if and only if (i_1), (i_2) and (ii_1) hold:
  (ii_1) There is an automaton $\mathbf{A} \in \mathscr{K}$ with $U_2|S(\mathbf{A})$.
(iii) $\mathscr{K}_0(\mathscr{G}) \subseteq \mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})$ if and only if (i_1) and (i_2) hold.

Proof. We only prove the first statement. Assuming $\mathscr{K}_3(\mathscr{G}) \subseteq \mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})$ also $\mathscr{K}_3(\mathscr{G}) \subseteq \mathbf{HS}^*\mathbf{P}^*_{\alpha_0}(\mathscr{K})$. Thus (i_1) follows from the Krohn—Rhodes Decomposition Theorem. Let G be a nontrivial abelian simple group in $\mathscr{G}$, say $G = Z_p$, the cyclic group of order p. Let H be any nonabelian p-group. We have $\mathrm{Aut}(H) \in \mathscr{K}_3(\mathscr{G})$ from the Krohn—Rhodes Decomposition Theorem. Thus also $\mathrm{Aut}(H) \in \mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})$ and, henceforth, there is an automaton $\mathbf{B} \in \mathbf{P}_{\alpha_0}(\mathscr{K})$ with $H|S(\mathbf{B})$. But then $H' \| S(\mathbf{B})$ follows from Proposition 3.8. Since H' is a nontrivial p-group we have $Z_p | H'$. Since $H' \| S(\mathbf{B})$ also $Z_p \| S(\mathbf{B})$ and, by Proposition 3.11, $Z_p \| S(\mathbf{A})$ for some $\mathbf{A} \in \mathscr{K}$. Thus (i_2) is satisfied by $\mathscr{K}$. To see that (i_3) holds, let $\mathbf{A}_0 = (A_0, X, \delta)$ be an automaton in $\mathscr{K}_3(\mathscr{G})$ with $U_3|S(\mathbf{A}_0)$ and such that none of the transformations $x^{\mathbf{A}_0}$, $x \in X$, is the identical mapping $A_0 \to A_0$. Since $\mathbf{A}_0 \in \mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})$, the above property yields $\mathbf{A}_0 \in \mathbf{HS}^+\mathbf{P}_{\alpha_0}(\mathscr{K}) \subseteq \mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K})$. The Krohn—Rhodes Decomposition Theorem implies $U_3|S(\mathbf{A})$ for some $\mathbf{A} \in \mathscr{K}$. This ends the proof of the necessity.

Conversely the assumptions (i_1)—(i_3), Corollary 3.9 and Proposition 3.10 imply that for every $G \in \mathscr{G}$ there is $\mathbf{A} \in \mathscr{K}$ with $G \| S(\mathbf{A})$ and similarly for U_3. Apply Corollary 3.3.

Corollary 3.13. For each $i = 0, 2, 3$, $\mathscr{K}_i(\mathscr{G}) \subseteq \mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})$ if and only if $\mathscr{K}_i(\mathscr{G}) \subseteq \mathbf{HS}^+\mathbf{P}_{\alpha_0}(\mathscr{K})$.

Corollary 3.14. The following are equivalent for a class $\mathscr{K}$ of automata:
(i) $\mathbf{HS}^*\mathbf{P}_{\alpha_0}(\mathscr{K})$ is the class of all automata.
(ii) $\mathbf{HS}^+\mathbf{P}_{\alpha_0}(\mathscr{K})$ is the class of all automata.
(iii) $\mathbf{HS}^+\mathbf{P}^+_{\alpha_0}(\mathscr{K})$ is the class of all automata.

Completeness criteria for the operator $\mathbf{HSP}_{\alpha_0}$ are formulated in [6, 3].

Abstract

A sufficient condition is given on a class $\mathscr{K}$ of automata ensuring that an automaton be homomorphically simulated by a generalized α_0-product (loop-free product) over $\mathscr{K}$ if and only if it is homomorphically simulated by an α_0-product with components in $\mathscr{K}$. As an application it is proved that a class $\mathscr{K}$ of automata is complete with respect to the homomorphic simulation by generalized α_0-products if and only if it is complete with respect to the homomorphic simulation by α_0-products, as far as nonempty words are considered.

MATHEMATICAL INSTITUTE, L. KOSSUTH UNIVERSITY
DEBRECEN, EGYETEM TÉR 1., 4010 HUNGARY

BOLYAI INSTITUTE, A. JÓZSEF UNIVERSITY
SZEGED, ARADI V. TERE 1., 6720 HUNGARY

References

[1] M. A. ARBIB (Ed.), Algebraic Theory of Machines, *Languages and Semigroups, with a major contribution by K. Krohn and J. L. Rhodes*, Academic Press, New York, 1968.
[2] J. DÉNES and P. HERMANN, On the product of all elements in a finite group, *Ann. of Discrete Math.*, 15 (1982), 107—111.
[3] P. DÖMÖSI and Z. ÉSIK, Critical classes for the α_0-product, *Theoret. Comput. Sci.*, to appear.
[4] EILENBERG, S., Automata, Languages and Machines, v. B, *Academic Press*, New York, 1976.
[5] ÉSIK, Z., Results on homomorphic realization of automata by α_0-product, in preparation.
[6] ÉSIK, Z. and DÖMÖSI, P., Complete classes of automata for the α_0-product, *Theoret. Comput.* Sci., 47 (1986), 1—14.
[7] GÉCSEG, F., Products of Automata, *Springer Verlag*, 1986.

Received Dec. 20. 1987

On minimal autonomous partitions of directed graphs and some applications to automata theory

BORIS B. KLOSS

We are here concerned with a class of partitions which are similar to the well known cyclic partitions of Markov chains. Let $G=(V, E)$ be a directed graph with a non-empty (possibly infinite) vertex set V and a set of directed edges E. Consider partitions $\pi=\{B_\alpha\}$ of a graph G, where $\{B_\alpha\}$ is a family of disjoint non-empty subsets (or blocks) $B_\alpha \subseteq V$ and $\bigcup_\alpha B_\alpha = V$. A partition π is called autonomous if for every block B_α either $\delta(B_\alpha)$ is empty or $\delta(B_\alpha) \subseteq B_\beta$ for some block B_β. Here $\delta(B)$ denotes the set of all vertices which are reached in one step from $B \subseteq V$. By the minimal autonomous partition (m.a.p.) of a directed graph G we mean such autonomous partition which is a refinement of any autonomous partition of this graph. Denote the m.a.p. of G by $\pi_{\min}(G)$, or simply $\pi_{\min}$ when non confusion is possible. The intersection of all autonomous partitions of G is an autonomous partition which is equal to the m.a.p. of G. Thus, the m.a.p. is uniquely determined for every directed graph.

These partitions turned out to be a very useful tool for studying some properties of automata and much of the motivation for the work discussed here derives from attempts to describe a structure of automata which are stable to the input-induced errors. My attention to examining the m.a.p. was also called by the paper [1] of A. Ádám, who introduced the autonomous partitions under the name *P*-partitions and considered these partitions from the graphtheoretical point of view. The main result of A. Ádám lies in the following. Let ϱ be the relation on a graph G such that for each pair of vertices $v, u \in V$ we have $(v, u) \in \varrho$ if and only if v and u are reached in equal number of steps from some vertex $w \in V$, i.e. there exist two paths of equal length from w to v and from w to u. Then $\varrho^T = \pi_{\min}$ for every sink-free directed graph, where ϱ^T denotes the transitive closure of ϱ. (Here and elsewhere we do not distinguish between partitions and the corresponding equivalence relations.) The above statement we shall call A. Ádám's theorem on minimal autonomous partitions.

The purposes of our paper are:

1) to describe the structure of m.a.p. for various types of directed graphs; and

2) to demonstrate the possibility of applications of A. Ádám's theorem to automata theory.

Minimal autonomous partitions

In this section we describe the structure of m.a.p. for arbitrary directed graphs, for graphs with finitely many sinks, for sink-free and source-free graphs and for strongly connected graphs.

By o we denote the trivial partition such that every block of o is a singleton. If τ is any relation on a graph, then we denote by τ^a the following relation: for each two vertices $v, u \in V$ we have $(v, u) \in \tau^a$ if and only if either $(v, u) \in \tau$ or there exists a finite sequence of pairs $v_i, u_i \in V$, $i=1, \ldots, n$, such that $(v_1, u_1) \in \tau$, $v_i \in \delta(v_{i-1})$ and $u_i \in \delta(u_{i-1})$ for $i=2, \ldots, n$ and $v_n=v$, $u_n=u$. τ^a will be called the autonomous closure of τ. It will be observed that o^a is exactly the relation ϱ defined above. By τ^T we denote the transitive closure of τ, i.e. $(v, u) \in \tau^T$ iff there exists a finite sequence of vertices $v_1, \ldots, v_n$ such that $(v_{i-1}, v_i) \in \tau$ for $i=2, \ldots, n$ and $v_1=v$, $v_n=u$.

We begin with A. Ádám's theorem ([1], Propositions 5,6):

Theorem 1. For an arbitrary graph $o^{aT} \leqq \pi_{\min}$. For an arbitrary sink-free graph $o^{aT}=\pi_{\min}$.

Remark 1. Although A. Ádám [1] dealt only with finite connected graphs, his proof of this theorem is valid for arbitrary graphs.

We are going to generalize A. Ádám's theorem in the following way. Let sink (G) be the number of sinks of G and $o^{n \times (aT)}$ means $o^{aT \ldots aT}$, where aT is repeated n times. We shall first give some properties of the relations $o^{n \times (aT)}$. Put $o^{\infty(aT)} = \bigcup_{n=1}^{\infty} o^{n \times a(T)}$, i.e. $(v, u) \in o^{\infty(aT)}$ iff $(v, u) \in o^{n \times (aT)}$ for some $n \geqq 1$. Note that $o^{n \times (aT)} \leqq o^{(n+1) \times (aT)}$, for each $n \geqq 1$, hence $o^{\infty(aT)}$ is a partition. Furthermore, one easily verifies that the following pairs of factor-graphs are isomorphic:

$$G/\pi_{\min}(G) \sim [G/o^{aT}]/[\pi_{\min}(G/o^{aT})] \qquad (*)$$

$$G/o^{(n+1) \times (aT)} \sim [G/o^{aT}]/[o^{n \times (aT)}(G/o^{aT})] \qquad (**)$$

or (by induction) for each $n, i \geqq 1$

$$G/\pi_{\min}(G) \sim [G/o^{i \times (aT)}]/[\pi_{\min}(G/o^{i \times aT})] \qquad (*)$$

$$G/o^{(n+1) \times (aT)} \sim [G/o^{i \times (aT)}]/[o^{n \times (aT)}(G/o^{i \times (aT)})] \qquad (**)$$

We are now in a position to prove the generalization of A. Ádám's theorem:

Theorem 2. For an arbitrary graph, $o^{\infty(aT)}=\pi_{\min}$. If sink $(G) \leqq n$, then $o^{(n+1) \times (aT)}=\pi_{\min}$.

Proof. 1) It follows from A. Ádám's theorem that $o^{n \times (aT)} \leqq \pi_{\min}$ for each $n \geqq 1$. Thus $o^{\infty(aT)} \leqq \pi_{\min}$. To prove $o^{\infty(aT)}=\pi_{\min}$, fix two vertices $(v, u) \in o^{\infty(aT)}$. We then have $(v, u) \in o^{k \times (aT)}$, for some $k \geqq 1$. Consequently, $(v', u') \in o^{(k+1) \times (aT)}$ if $v' \in \delta(v)$ and $u' \in \delta(u)$. From this it follows that $o^{\infty(aT)}$ is an autonomous partition. Suppose $o^{\infty(aT)} \neq \pi_{\min}$. Then $o^{\infty(aT)}$ is a proper refinement of $\pi_{\min}$ and the minimality of $\pi_{\min}$ gives the contradiction.

2) Now prove the second assertion.
Induction base: If sink $(G)=0$, then $o^{aT}=\pi_{\min}$ by A. Ádám's theorem.
Induction step: If sink $(G)=n+1$ and $o^{aT}\neq\pi_{\min}$, then sink $(G/o^{aT})\leqq n$. Indeed, if $o^{aT}\neq\pi_{\min}$, then there exist blocks A, B, C of o^{aT} with $\delta(A)\cap B\neq\emptyset$, $\delta(A)\cap C\neq\emptyset$ and $B\cap C=\emptyset$. (Here $\emptyset$ denotes the empty set.) This means that there is a sink in A, but A, considered as a vertex of the factor-graph G/o^{aT}, is not a sink. Thus sink $(G/o^{aT})\leqq n$.

Suppose $o^{(n+1)\times(aT)}=\pi_{\min}$ holds for each graph G' with sink $(G')\leqq n$. Then

$$G'/\pi_{\min}(G')\sim G'/o^{(n+1)\times(aT)}(G').$$

where $G'=G/o^{aT}$ and $\sim$ means graph isomorphism. On the other hand, properties $(*)$ and $(**)$ give

$$G'/\pi_{\min}(G')\sim G/\pi_{\min}(G)$$

$$G'/o^{(n+1)\times(aT)}(G')\sim G/o^{(n+2)\times(aT)}(G).$$

Thus

$$G/\pi_{\min}(G)\sim G/o^{(n+2)\times(aT)}(G)$$

and consequently, $\pi_{\min}=o^{(n+2)\times(aT)}$ if sink $(G)=n+1$. Q.E.D.

Examples. Fig. 1 shows a graph G with sink $(G)=1$, $o^{aTa}\neq\pi_{\min}$, $o^{aTaT}=\pi_{min}$. Fig. 2 shows a graph G with sink $(G)=2$, $o^{aTaTa}\neq\pi_{\min}$, $o^{aTaTaT}=\pi_{\min}$. For the graph in Fig. 3 we have sink $(G)=3$, $o^{aTaTaTa}\neq\pi_{\min}$, $o^{aTaTaTaT}=\pi_{\min}$.

These examples give rise to the following

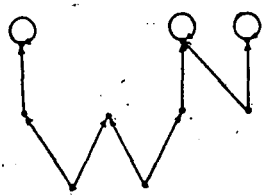

Fig. 1.

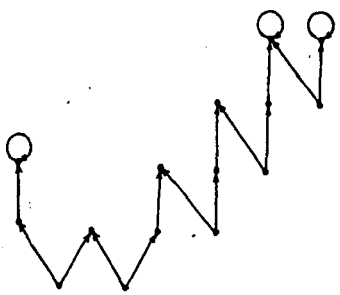

Fig. 2.

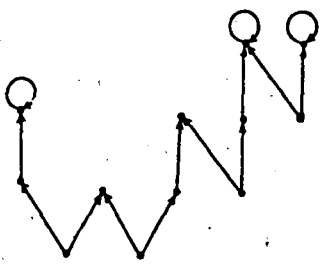

Fig. 3.

Proposition 1. For each integer n there exists a graph G such that sink $(G)=n$ and $o^{n\times(aT)a}\neq\pi_{\min}$.

Question 1. Does Proposition 1 remain valid when we restrict ourselves to finite graphs without sources?

Question 2. For any n, characterize the graphs having exactly n sinks such that $o^{n\times(aT)a}=\pi_{\min}$ holds in Theorem 2.

Corollary 1. If G is finite, then $o^{n\times(aT)}=\pi_{\min}$ for some integer $n\leqq|V|$.

Proof. If every vertex of G is a sink, then $o=\pi_{\min}$. Elsewise, sink $(G)\leqq|V|-1$ and we can apply Theorem 2.

Question 3. What is the smallest number $f(k)$ such that $o^{f(k)\times(aT)}=\pi_{\min}$ for every finite graph with $|V|=k$?

Now let us consider sink-free graphs. The relation o^a is not transitive, in general, even if a graph has no sink and no source (see Fig. 4). (This example also provides a particular answer to Problem 3 in [1].) But for strongly connected graphs the relation o^a is always transitive.

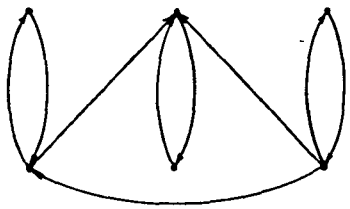

Fig. 4.

Proposition 2. For an arbitrary strongly connected graph, $o^a=\pi_{\min}$.

Proof. First, let G be a finite strongly connected graph. Let $B\in\pi_{\min}$ be an arbitrary block of its m.a.p. and let $c\in B$ be a vertex in this block. Consider the factor graph $G/\pi_{\min}$. Obviously, $G/\pi_{\min}$ is a cycle. Denote its length by p. Consider a sequence of sets $S_k=\delta^{kp}(c)$, $k=0, 1, 2, \ldots$, where $\delta^n(c)=\delta(\delta^{n-1}(c))$, $\delta^0(c)=c$, $\delta^1(c)=\delta(c)$. Note that $S_k\subseteqq B\in\pi_{\min}$ and for every pair of vertices $v, u\in S_k$ we have $(v, u)\in o^a$, for each $k=0, 1, 2, \ldots$. Since G is finite, the sequence S_k becomes stationary, i.e. there exist integers $l\leqq m$ such that $\delta^p(S_l)=S_{l+1}$, $\delta^p(S_{l+1})=S_{l+2}, \ldots, \delta^p(S_m)=S_l$. Since G is strongly connected, $\bigcup_{i=1}^{m} S_i=B$. If $l=m$, then the proposition is already proved (in this case $S_l=B$ and $(v, u)\in o^a$ for each pair $v, u\in B$). If not, suppose without the loss of generality that all sets in the system $S=\{S_l, \ldots, S_m\}$ are different. A family of sets $\{S_\alpha\}\subseteqq S$, $\alpha\in A\subseteqq\{l, \ldots, m\}$, will be called a maximal system if and only if $\bigcap_{\alpha\in A} S_\alpha=\tilde{S}\neq\emptyset$ and for each n $(l\leqq n\leqq m)$ such that $n\notin A$ we have $\tilde{S}\cap S_n=\emptyset$. Let $\tilde{S}=\{\tilde{S}_1, \ldots, \tilde{S}_q\}$ be the family of intersections of the maximal systems. It is not difficult to see that $\delta^p(\tilde{S}_1)\subseteqq\tilde{S}_2, \ldots, \delta^p(\tilde{S}_q)\subseteqq\tilde{S}_1$ for a

suitable numeration of the sets $\tilde{S}$. Furthermore, if $i \neq j$, then $\tilde{S}_i \cap \tilde{S}_j = \emptyset$. Since G is strongly connected, one has $\bigcup_{i=1}^{q} S_1 = B$. Therefore we can consider the following partition:

$$\pi = \{\tilde{S}_1, \delta^1(\tilde{S}_1), \ldots, \delta^{p-1}(\tilde{S}_1), \tilde{S}_2, \ldots, \delta^{p-1}(\tilde{S}_q)\}.$$

Obviously, π is autonomous partition and π is a proper refinement of $\pi_{\min}$. This contradicts the minimality of $\pi_{\min}$. Hence $l=m$ and $o^a=\pi_{\min}$.

The general case, when G is an arbitrary strongly connected graph, is reducible to the previous one. Indeed, let B be an arbitrary block of the m.a.p. We are going to show that $(v, u) \in o^a$ for each pair of vertices $v, u \in B$. Since a strongly connected graph has no sink, therefore it fulfils the suppositions of A. Ádám's theorem. Hence $(v, u) \in o^{aT}$. This means that there exists a sequence of vetrices $v_1, \ldots, v_n \in B$ such that $v_1 = v$, $v_n = u$ and $(v_i, v_{i+1}) \in o^a$ for $i=1, \ldots, n-1$. Select two paths of equal length from v_i to v_i and from v_i to v_{i+1} for each $i=1, \ldots, n-1$ and take an arbitrary path from v_n to v_1. Consider the subgraph G' of G consisting all vertices and edges of selected paths. It is clear that G' is a finite strongly connected graph. Moreover, the vertices v and u belong to the same block of $\pi_{\min}(G')$. Consequently, $(v, u) \in o^a$ by the previous part of the proof (note that $o^a(G')$ is the refinement of $o^a(G)$). Q.E.D.

Remark 2. In addition, strongly connected graphs have another advantageous property: it is a well known fact in the theory of Markov chains that for such graphs the equality $p=p^*$ holds (see below).

Now we are going to generalize Proposition 2. When does $o^a=\pi_{\min}$ hold for sink-free and source-free graphs? This problem is closely related to Problem 2 in [1]: when is the length p of the cycle of the functional graph $G/\pi_{\min}$ equal to the greatest common divisor p^* of all cycle lengths of G? Let $\tilde{p}$ be the greatest common divisor of all cycle lengths of the induced subgraph spanned by all generators of G, i.e. $\tilde{p}$=g.c.d. {length (C): every vertex of cycle C is a generator of G}. (A vertex v is called generator if for each vertex u there exists a path from v to u.)

One has the following

Theorem 3. If a finite connected graph G has no source, then $o^a=\pi_{\min}$ iff there exists at least one generator of G and $p=\tilde{p}$.

Proof. Assume that $o^a=\pi_{\min}$ and G has no source. Then o^a is a transitive relation. It is not difficult to see that there exists a generator v of G. Let u be a vertex such that there is a path from u to v of length p. If $v=u$, then $p=\tilde{p}$. Otherwise, since the vertices v and u belong to the same block of $\pi_{\min}$ (therefore $(v, u) \in o^a$) and since v is a generator, there exist two paths from v to v and from v to u of equal length kp (for some integer $k \geqq 1$). It is clear that one can find two cycles, both containing v, with lengths kp and $kp+p$. Hence $p=\tilde{p}$.

Now let $p=\tilde{p}$ and suppose that there exists a generator v of G. Then there are two cycles C_1 and C_2 such that $v \in C_1$, $v \in C_2$ and the greatest common divisor of l_1=length (C_1) and l_2=length (C_2) equals p. Indeed, the subgraph $\tilde{G}$ of G spanned by all generators is strongly connected, hence we can apply Proposition 2, our assumption $p=\tilde{p}$, then Remark 2 and the construction of the previous part of this proof. It is clear that if $(v_1, v_2) \in o^a$, then $(v_1, v_2) \in \pi_{\min}$. We are going to show the converse implication. Let v_1 and v_2 be arbitrary vertices which belong to the same block of

$\pi_{\min}$. Since v is a generator of G, there exist two paths from v to v_1 (of the length m_1) and from v to v_2 (of the length m_2). It should be observed that $l_1=k_1p$, $l_2=k_2p$, $|m_1-m_2|=k_3p$, for some $k_1\geqq 1$, $k_2\geqq 1$, $k_3\geqq 0$. From the fact that the equation

$$l_1x+l_2y=|m_1-m_2|$$

is solvable in integers, it follows that there exist two paths of equal length from v to v_1 and from v to v_2, respectively. Q.E.D.

Corollary 1. If a finite connected graph G has no source, then $o^a=\pi_{\min}$ implies $p=p^*$.

Proof. The divisibility relations $p|p^*$ and $p^*|\tilde{p}$ are clear. If $p=\tilde{p}$, then $p=p^*$.

The converse implication in Corollary 1 is not valid, in general (see Fig. 5).

We finish this section with the remark that the above results were not intended as an overview of the m.a.p. and Problems 1—2 proposed by A. Ádám [1] are still open.

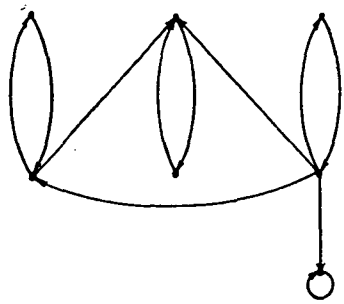

Fig. 5.

Applications to automata theory

In this section we are going to describe a class of automata which are stable to the input-induced errors. First introduce some notations used below. By automaton we mean a system $A=(X, S, \delta)$, where X and S are arbitrary finite non-empty sets, called the input alphabet and the state set, respectively, and δ: $S\times X\to S$ is called the transition function. By δ we also denote the natural extension of the transition function to a mapping $2^S\times X^*\to 2^S$, where 2^S is a family of subsets of S and X^* is a free monoid generated by X. By the m.a.p. of an automaton we mean the m.a.p. of its transition diagram. A block B of the m.a.p. of an automaton is called cyclic if $\delta(B, J)\subseteqq B$ for some non-empty $J\in X^*$. The set of states which belong to the cyclic blocks is denoted $C(S)$. By the period of automaton A we mean the least common multiple of all cycle lengths of the transition diagram of the factor automaton $A/\pi_{\min}$. We denote the period of A by $p(A)$. If $A_1=(X, S_1, \delta_1)$ and $A_2=(X, S_2, \delta_2)$ are automata with $S_1\cap S_2=\emptyset$ and the same input alphabet, then $A_1\times A_2=$ $=(X, S_1\times S_2, \delta)$, where $\delta((s_1, s_2), x)=(\delta_1(s_1, x), \delta_2(s_2, x))$ for each $s_1\in S_1$, $s_2\in S_2$

and $x \in X$, is called the product of the automata and $A_1 + A_2 = (X, S_1 \cup S_2, \delta)$, where

$$\delta(s, x) = \begin{cases} \delta_1(s, x), & \text{if } s \in S_1 \\ \delta_2(s, x), & \text{if } s \in S_2, \end{cases}$$

is called the sum of the automata. An automaton $\tilde{A} = (X, \tilde{S}, \tilde{\delta})$ is called a subautomaton of $A = (X, S, \delta)$ if $\tilde{S} \subseteq S$ and $\tilde{\delta}(\tilde{s}, x) = \delta(\tilde{s}, x)$ for every choice of $\tilde{s} \in \tilde{S}$, $x \in X$. An automaton A is said to be strongly connected if for every pair of states $s, t \in S$ there are such words $J_1, J_2 \in X^*$ that $\delta(s, J_1) = t$ and $\delta(t, J_2) = s$. In other words, an automaton A is strongly connected iff the transition diagram of A is strongly connected. An automaton is said to be connected if its transition diagram, considered as a non-oriented graph, is connected. Note that every automaton A is a sum of connected automata $A = \sum_{\alpha} A_\alpha$. We say that an automaton A can be represented by a parallel composition of automata B and C if there exists a subautomaton D of $B \times C$ such that A is a homomorphic image of D. The onto mapping $h: S' \to S$ is called a homomorphism from $D = (X, S', \delta')$ to $A = (X, S, \delta)$ if $\delta(h(s), x) = h(\delta'(s, x))$ for every choice of $s \in S'$, $x \in X$.

An automaton A is called autonomous if $\delta(s, x) = \delta(s, y)$ for each $s \in S$ and $x, y \in X$. It should be observed that an automaton A is autonomous iff it is isomorphic to the factor automaton $A/\pi_{\min}$.

An automaton A is called to be directable (or cofinal) if there exists a word $J \in X^*$ such that $|\delta(S, J)| = 1$, where $|\cdot|$ denotes the cardinality. Such words J are called directing.

A directable automaton is called definite if there exists an integer n such that every word, whose length is greater than or equal to n, is directing.

The automata we will be concerned with belong to a class defined by the following properties.

Definition 1. An automaton $A = (X; S, \delta)$ is called correctable if there exists $J \in X^*$ such that $\delta(s, J_1 J) = \delta(s, J_2 J)$ for every state $s \in S$ and every two words $J_1, J_2 \in X^*$ of equal length. Such words J are called correcting.

Note that it is just the case of S. Winograd's automata which are synchronized with probability 1 with respect to the input-induced errors [6].

The automata of this type are capable of "forgetting" all previously occurred errors after accepting a specially selected correcting sequence of inputs. This provides the advantages of their use in technique.

The next assertion follows immediately from A. Ádám's theorem.

Correctability Criterion. An automaton A is correctable iff there exists a (correcting) word $J \in X^*$ such that $|\delta(B, J)| = 1$ for each block $B \in \pi_{\min}(A)$.

This criterion allows us to describe the structure of correctable automata more precisely.

First Decomposition Theorem. An automaton A is correctable iff it can be represented by a parallel composition of autonomous and directable automata.

Sketch of the proof. The following five lemmas imply the sufficiency of the theorem.

The next assertion is obvious:

Lemma 1. Every autonomous automaton is correctable.

Lemma 2. An automaton A is directable iff the following three conditions are fulfilled:

A is correctable,
A is connected,
$p(A)=1$.

Proof. Suppose that $p(A)=1$ holds for a connected correctable automaton A. Then $A/\pi_{\min}$ has only one cycle and this cycle is a loop. Denote by B the set of states $s\in S$ of A such that the natural homomorphism γ of A onto $A/\pi_{\min}$ carries s to the unique cyclic vertex of $A/\pi_{\min}$. It is easy to see that there exists a natural number n such that $\delta(S, I)\subseteq B$ whenever the length of the word I is at least n. The Correctability Criterion implies the existence of a word $J_1\in X^*$ such that $|\delta(B, J)|=1$. Let $J_2\in X^*$ be an arbitrary word whose length is at least n and let J be defined by $J=J_2J_1$. Then J is a directing word.

Conversely, assume that A is a directable automaton. Obviously, A is connected and correctable. Our last aim is to verify $p(A)=1$. We shall show that $p(A)>1$ leads to a contradiction. Indeed, $p(A)>1$ implies the existence of two states $s, t\in S$ such that s and t belong to different cyclic blocks of $\pi_{\min}$. Thus $\delta(s, J)\neq\delta(t, J)$ for every word $J\in X^*$ and the lemma follows.

Lemma 3. ([4]). A product of finitely many correctable automata is correctable.

Lemma 4. ([4]). Every subautomaton of a correctable one is correctable.

Lemma 5. Every homomorphic image (consequently every factor automaton) of a correctable automaton is correctable.

Proof. Consider a homomorphism $h: A\to B$, where A is correctable. Let us start with three states s_1, s_2, s_3 of B such that $\delta_B(s_1, J_1)=s_2$, $\delta_B(s_1, J_2)=s_3$ with some words J_1, J_2 which are of equal length. (Here δ_B means the transition function of B and δ_A denotes the transition function of A). Then obviously

$$\delta_A(s_1', J_1)\in h^{-1}(s_2), \quad \delta_A(s_1', J_2)\in h^{-1}(s_3)$$

for an arbitrary element s_1' of $h^{-1}(s_1)$; thus the correcting word J of A fulfils

$$\delta_A(s_1', J_1J) = \delta_A(s_1', J_2J).$$

Hence

$$\delta_B(s_2, J) = \delta_B(s_3, J)$$

and the lemma follows.

Now we are going to prove the necessity. Let A be a correctable automaton. Consider the following three cases.

1. Let A be strongly connected. Then the partition classes mod $\pi_{\min}(A)$ can be denoted by $B_1, B_2, \ldots, B_q$ in such a manner that $\delta(B_i, x)\subseteq B_{i+1}$ if $1\leqq i\leqq q-1$ and $\delta(B_q, x)\subseteq B_1$ (for each $x\in X$). Let us choose a set $C=\{s_1, s_2, \ldots, s_q\}$ such that $s_i\in B_i$ for each i $(1\leqq i\leqq q)$.

Consider the family of all sets $\delta(C, J)$ where (C is fixed and) J runs through all the elements of X^*. Since A is a finite automaton, this family consists of a finite number of different members. Denote the members of the family by $C_1, C_2, \ldots, C_n$ (the

ordering is arbitrary). $C_1, C_2, \ldots, C_n$ are pairwise different (but not necessarily disjoint) state sets and their union $C_1 \cup \ldots \cup C_n$ equals S (otherwise we could get a contradiction to the strongly connectedness of A). It is easy to see that $|C_i \cap B| = 1$ for every choice of C_i and B, where $1 \leqq i \leqq n$ and B is a block mod $\pi_{\min}(A)$. For every choice of C_i and $x \in X$ there exists a unique C_j such that $\delta(C_i, x) = C_j$ $(1 \leqq i \leqq n$, $1 \leqq j \leqq n)$.

Consider the automaton $A_d = (X, \{C_i\}, \delta_d)$, where $\delta_d(C_i, x) = C_j$ iff $\delta(C_i, x) = C_j$. Denote $A_a = A/\pi_{\min}$. It is not difficult to see that A_d is a directable automaton. This assertion follows from the Correctability Criterion. Indeed, since A is correctable, then there exists a correcting word $J \in X^*$ such that $|\delta(B, J)| = 1$ for each block B mod $\pi_{\min}(A)$. Consider two arbitrary states C_i, C_j of A_d. Let $C_i = \{s_1, \ldots, s_q\}$ and $C_j = \{s'_1, \ldots, s'_q\}$, where $s_\alpha \in B_\alpha$, $s'_\alpha \in B_\alpha$. Since s_α and s'_α belong to the same block mod $\pi_{\min}(A)$, we have $\delta(s_\alpha, J) = \delta(s'_\alpha, J)$ for each α $(1 \leqq \alpha \leqq q)$. Put $s''_\alpha = \delta(s_\alpha, J)$. Obviously, $\{s''_1, \ldots, s''_q\}$ is a member of the family $C = \{C_1, \ldots, C_n\}$. Let $\{s''_1, \ldots, s''_q\} = C_k$ where $1 \leqq k \leqq q$. Then

$$\delta_d(C_i, J) = \delta_d(C_j, J) = C_k,$$

hence A_d is directable. Obviously, A_a is autonomous. The mapping from $A_a \times A_d$ to A taking a pair of states $B \in \pi_{\min}(A)$ (this is a state of A_a) and $C_i \in C$ (this is a state if A_d) into the state $s = B \cap C_i$ of A is a required homomorphism.

2. Let A be connected.

Lemma 6. ([4]). Every connected correctable automaton contains a unique strongly connected subautomaton (which is evidently correctable).

Remark 3. It will be noted that Lemma 6 is not valid in general for infinite automata.

Denote by $\tilde{A} = (X, \tilde{S}, \tilde{\delta})$ the strongly connected subautomaton of A. Let $\tilde{A}_a = \tilde{A}/\pi_{\min}(\tilde{A})$ and $\tilde{A}_d = (X, \{\tilde{C}_i\}, \tilde{\delta}_d)$, $\tilde{C}_i \subseteqq \tilde{S}$, be autonomous and directable components of $\tilde{A}$, constructed analogously to the previous part of this proof, i.e. $\tilde{A}$ can be represented by a parallel composition of $\tilde{A}_a$ and $\tilde{A}_d$. Consider the automaton $A_d = (X, (S \setminus \tilde{S}) \cup \{C_i\}, \delta_d)$, where

$$\delta_d(b, x) = \begin{cases} \delta(b, x), & \text{if } b \in S \setminus \tilde{S} \text{ and } \delta(b, x) \in S \setminus \tilde{S}; \\ \text{arbitrary } \tilde{C}_i \in \{\tilde{C}_i\} \text{ such that } \delta(b, x) \in \tilde{C}_i, & \text{if } b \in S \setminus \tilde{S} \text{ and } \delta(b, x) \in \tilde{S}; \\ \tilde{\delta}_d(b, x), & \text{if } b \in \{\tilde{C}_i\}. \end{cases}$$

The automaton A_d is directable. Indeed, it is easy to see that there exists a word $J_1 \in X^*$ such that $\delta_d((S \setminus \tilde{S}) \cup \{\tilde{C}_i\}, J_1) \subseteqq \{\tilde{C}_i\}$. Let $J_2 \in X^*$ be a correcting word of $\tilde{A}$, hence J_2 is a directing word of $\tilde{A}_d$. Let J be defined by $J = J_1 J_2$. Then J is a directing word of A_d. Denote $A_a = A/\pi_{\min}(A)$. Our last aim is to find a subauomtaton of $A_a \times A_d$ which can be mapped homomorphically onto A. Consider the subautomaton $\bar{A}$ whose states are all the pairs (B, b), where $B \in \pi_{\min}(A)$, $b \in (S \setminus \tilde{S}) \cup \{\tilde{C}_i\}$, such that $B \cap b \neq \emptyset$. Note that $\bar{A}$ really is a subautomaton. Then the mapping $(B, b) \to B \cap b$ is a required homomorphism.

3. Let A be an arbitrary correctable automaton. Represent A in the form $A = \sum_\alpha A_\alpha$, where A_α is connected for each α.

Lemma 7. ([4]). Suppose $A=\sum_{i=1}^{n} A_i$; then A is correctable iff A_i is correctable for each $i=1, \ldots, n$. If A is correctable and $A=\sum_{\alpha} A_\alpha$ (where there is an infinity of summands), then A_α is correctable for each α.

Let A_a^α and A_d^α be automous and directable components of A_α, constructed analogously to the first and second parts of our proof. Then A can be represented by a parallel composition of $\sum_{\alpha} A_a^\alpha$ and $\prod_{\alpha} A_d^\alpha$. It will be noted that a sum of autonomous automata is autonomous and a product of finitely many directable automata is directable. Since every correcting word of A is a directing word of $\prod_{\alpha} A_d^\alpha$, the automaton $\prod_{\alpha} A_d^\alpha$ is directable, even if there were an infinite number of multiplicands. Q.E.D.

Remark 4. It follows from part 1 of the previous proof that every strongly connected correctable automaton is a homomorphic image of a product of strongly connected autonomous and strongly connected directable automata.

A. Ádám's theorem can be applied to the description the other types of automata.

Definition 2. A correctable automaton A is called self-correctable if there exists an integer n such that every word, whose length is greater than or equal to n, is correcting.

The smallest n which satisfies the above condition is called the correction time and denoted by $n(A)$.

Self-correctability Criterion. An automaton A is self-correctable iff there exists an integer n such that $|\delta(B, J)| \doteq 1$ for each block $B \in \pi_{\min}(A)$ and each word $J \in X^n$. The smallest n which satisfies this condition equals $n(A)$.

Second Decomposition Theorem. An automaton A is self-correctable iff it can be represented by a parallel composition of autonomous and definite automata.

The proof might have been arranged analogously to the proof of First Decomposition Theorem, but we are here suggested a simpler way of proving this theorem.

First we establish some preliminary results on self-correctable automata.

The following result is obvious:

Lemma 8. An automaton A is self-correctable iff there exists an integer n such that the equality

$$\delta(s, I_1 J) = \delta(s, I_2 J)$$

holds for every choice of the state s and the words I_1, I_2, J where I_1, I_2 are of equal length and the length of J is n.

Suppose that A is self-correctable, let the smallest possible $n\,(=n(A))$ be considered (cf. Lemma 8). For every t $(\geqq n)$ we denote by $F_t(s, J)$ the state $\delta(s, IJ)$ where s is a state J is a word of length n and I is an abritrary word whose length is $t-n$.

Supplement to Lemma 8. The sequence of functions

$$F_n, F_{n+1}, F_{n+2}, \ldots \qquad (***)$$

is periodic.

Proof. The set of states of A and the set of words of length n are finite. Thus the sequence $(***)$ contains only a finite number of different members. Let q be the smallest number such that there is a number t_0 for which $n \leqq t_0 < q$ and $F_{t_0} = F_q$ are valid. We can show without difficulty that $F_t = F_{t'}$ implies $F_{t+1} = F_{t'+1}$. Consequently, the sequence $(***)$ is periodic; the length of its period and pre-period are $p = q - t_0$ and t_0, respectively. (In other words: $F_t = F_{t'}$ if and only if $t \geqq t_0$, $t' \geqq t_0$ and $t \equiv t' \pmod{p}$ are true.)

Remark 5. The period p of $(***)$ equals the period $p(A)$ of the automaton A.

Proof of the Second Decomposition Theorem. Since Lemmas 1—7 are valid for self-correctable automata after replacing the words correctable by self-correctable and directable by definite, then a composition of autonomous and definite automata is self-correctable. Now let A be a self-correctable automaton. Then one can consider the definite component of A as a connection of storage device on a shift register (for preservation of last $n(A)$ inputs) and a set of $p(A)$ devices for computing functions $F_{t_0}, \ldots, F_{t_0+p(A)-1}$. It is easy to see that the definite component is a definite in fact automaton. In this case the autonomous component $A/\pi_{\min}$ determines the function which value corresponds to the present state of A. Q.E.D.

Remark 6. One can show that every strongly connected self-correctable automaton is a homomorphic image of a product of strongly connected autonomous and strongly connected definite automata (cf. Remark 4). M. Ito and J. Duske proved in [3] that every strongly connected definite automaton is a homomorphic image of a shift register. (Recall that a shift register in [3] is an automaton (X, X^n, δ) where X is finite, $n \geqq 1$ and $\delta((x_1, \ldots, x_n), x) = (x_2, \ldots, x_n, x)$ for every choice of $x \in X$, $x_i \in X$, $i = 1, \ldots, n$.) Thus, every strongly connected self-correctable automaton is a homomorphic image of a product of strongly connected autonomous automaton and a shift register.

Now let us estimate the correction time $n(A)$ of self-correctable automata.

Theorem 4. Let A be a self-correctable automaton and let k $(\geqq 0)$ be the smallest number such that each $J \in X^{n(A)+k}$ satisfies:

1) $\delta(S, J) \subseteq C(S)$; and

2) if $\delta(B_1, J) \subseteq B$ and $\delta(B_2, J) \subseteq B$ for some $B_1, B_2, B \in \pi_{\min}(A)$, then

$$\delta(B_1, J) = \delta(B_2, J).$$

Then

$$n(A) + k \leqq |S| - m,$$

where m is the maximum of all cycle lengths of the transition diagram of $A/\pi_{\min}$. If A is a strongly connected self-correctable automaton, then

$$p(A) \times |X|^{n(A)} \geqq |S|.$$

Remark 7. In particular, if A is a connected self-correctable automaton, then

$$n(A) + k \leqq |S| - p(A).$$

Remark 8. Since $J \in X^{n(A)+k}$ and $k \geqq 0$, then $|\delta(B_i, J)| = 1$ in Theorem 4 $(i = 1, 2)$. Therefore one can write $\delta(B_i, J) \in B$ instead of

$$\delta(B_i, J) \subseteq B.$$

Proof of Theorem 4. Let us first suppose that A is a strongly connected self-correctable automaton. Since $C(S)=S$, then $k=0$ in this case. Given an integer n consider the relation

$$P_n = \{(s_1, s_2)\in S^2\colon \delta(s_1, J) = \delta(s_2, J) \text{ for each } J\in X^n\}.$$

It is clear that $P_0=\{(s, s)\colon s\in S\}$. Since A is strongly connected, then $P_n\supseteq\pi_{\min}(A)$ holds for each $n\geqq 0$. It follows from Self-correctability Criterion that $P_{n(A)}\supseteq \supseteq\pi_{\min}(A)$ where $n(A)$ is the correction time of A. Thus, $P_{n(A)}=\pi_{\min}(A)$. Moreover, $P_n\subseteq P_{n+1}$ and $P_n\neq P_{n+1}$ iff $n<n(A)$. The strong connectedness of A implies that the transition diagram of $A/\pi_{\min}$ is a cycle. Therefore the number of blocks of $\pi_{\min}(A)$ is equal to the period $p(A)$ of A. Obviously, the number of blocks of $P_{n(A)-1}$ is greater than or equal to $p(A)+1$. Similarly, if $0\leqq i\leqq n(A)$ then the number of blocks of $P_{n(A)-i}\geqq p(A)+i$. In particular, with $i=n(A)$, we get

$$n(A)\leqq |S|-p(A).$$

Now consider an arbitrary self-correctable automaton A. Without the loss of generality we may restrict ourselves to the case where A is connected.

The proof of Theorem 4 will be continued after verifying a lemma.

Lemma 9. Let A be a connected self-correctable automaton. Then there exists a partition $\pi=\{B_i\}$, $i=1, \ldots, n$, of A such that

1) for any i $(1\leqq i\leqq n)$, the elements of $B_1\cup B_2\cup\ldots\cup B_i$ form a subautomaton of A; moreover, (X, B_1, δ) is strongly connected and self-correctable;

2) if $1\leqq i\leqq n$ and J is a word whose length is denoted by l, then $\delta(B_i, J)\subseteq \subseteq B_1\cup\ldots\cup B_{i-l^*}$, where $l^*=\min(l, i-1)$.

Proof. One can choose (using Lemma 6 and the remark at the beginning of the proof of the Second Decomposition Theorem) the unique selfcorrectable strongly connected subautomaton (X, B_1, δ) of A.

Consider a sequence of sets:

$$B_2 = \{s\in S\setminus B_1\colon \delta(s, x)\in B_1 \text{ for each } x\in X\};$$

$$B_3 = \{s\in S\setminus(B_1\cup B_2);\ \delta(s, x)\in B_1\cup B_2 \text{ for each } x\in X\};$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$B_i = \{s\in S\setminus(B_1\cup\ldots\cup B_{i-1})\colon \delta(s, x)\in B_1\cup\ldots\cup B_{i-1} \text{ for each } x\in X\};$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

Clearly, one can find an integer m $(\geqq 1)$ such that $B_i=\emptyset$ iff $i>m$. Since the family of disjoint sets $\{B_i\}$, $i=1, \ldots m$, satisfies the conditions $1-2$, we only have to prove that $\{B_i\}$ is really a partition, i.e. $\bigcup_{i=1}^{m} B_i=S$. Put $\tilde{S}=S\setminus(B_1\cup\ldots\cup B_m)$. We are going to show that $\tilde{S}=\emptyset$.

Assume now that $\tilde{S}\neq\emptyset$. We derive a contradiction from this assumption. It is easy to see that there exists $r\geqq 1$ such that $\delta(S, J)\subseteq C(S)$ for all words $J\in X^r$. Using the definition of $\tilde{S}$ one also easily obtains that for any $s\in\tilde{S}$ there exists a sequence of words $J_i^s\in X^i$ such that $\delta(s, J_i^s)\in\tilde{S}$ where $i=1, 2, \ldots$. Let $\tilde{s}$ be an arbi-

trary state of $\tilde{S}$, then $s=\delta(\tilde{s}, J_r^s)$ belongs to a cyclic block of $\pi_{\min}(A)$. A moment's consideration shows that there exists a state t of B_1 which belongs to the same block of the m.a.p. Indeed, let the cyclic blocks of $\pi_{\min}(A)$ be denoted by $C_1, C_2, \ldots, C_q$. Since (X, B_1, δ) is a subautomaton of the connected automaton A, $B_1 \cap C_j \neq \emptyset$ for any $j=1, \ldots, q$. Now let $s \in C_j$, we choose an arbitrary state t of $B_1 \cap C_j$. Thus, the states $s \in \tilde{S}$ and $t \in B_1$ belong to the same (cyclic) block of $\pi_{\min}(A)$. Furthermore, $\delta(s, J_i^s) \in \tilde{S}$ for all $i=1, 2, \ldots$ and $\delta(t, J) \in B_1$ for each word $J \in X^*$. Obviously, $B_1 \cap \tilde{S} = \emptyset$. Therefore $\delta(s, J_i^s) \neq \delta(t, J_i^s)$ for every choice of $i=1, 2, \ldots$ (Note that the length of J_i^s equals i). This contradicts the Self-correctability Criterion and the lemma follows.

Proof of Theorem 4 (final part). Now (using Lemma 9) let us select the strongly connected subautomaton $\tilde{A}=(X, B_1, \delta)$ of A and let r ($\geqq 1$) be the smallest number which satisfies $\delta(S, J) \subseteq B_1$ for all $J \in X^r$. Clearly $\delta(S, J) \subseteq C(S)$ when $J \in X^r$ and it follows from Lemma 9 that

$$r \leqq |S \setminus B_1|. \tag{1}$$

Since $\tilde{A}$ is strongly connected, therefore by the previous part of the proof one has

$$n(\tilde{A}) \leqq |B_1| - p(\tilde{A}). \tag{2}$$

It will be noted that

$$p(\tilde{A}) = p(A). \tag{3}$$

Also note that

$$n(A) + k \leqq n(\tilde{A}) + r. \tag{4}$$

Clearly (1), (2), (3) and (4) jointly imply

$$n(A) + k \leqq n(\tilde{A}) + r \leqq |B_1| - p(\tilde{A}) + |S \setminus B_1| = |S| - p(A),$$

and the first assertion of Theorem 4 is proved.

But it follows immediately from Remark 6 that

$$p(A) \times |X|^{n(A)} \geqq |S|$$

holds for strongly connected self-correctable automata. Q.E.D.

Now let A be definite, then the smallest n which satisfies:

$$|\delta(S, J)| = 1 \quad \text{for all} \quad J \in X^n$$

is called the degree of A and is denoted by $d(A)$.

Corollary 1. Let A be a definite automaton, then

$$d(A) \leqq |S| - 1.$$

If A is a strongly connected definite automaton, then

$$|X|^{d(A)} \geqq |S|.$$

Proof. If A is definite, then it is self-correctable. By Lemma 2 and remark at the beginning of the proof of the Second Decomposition Theorem one has $p(A)=1$. Since A is definite, it is connected, therefore the maximum m of all cycle lengths of the transition diagram of $A/\pi_{\min}$ equals $p(A)=1$. Finally we show that $d(A)=n(A)+k$ (cf. Theorem 4). It is clear that $d(A) \geqq n(A)+k$. Now let $J \in X^*$ be defined by $J=J_1J_2$ where the length of J_1 equals k and the length of J_2 equals $n(A)$. Then all the

states of $\delta(S, J_1)$ belong to the unique cyclic block of $\pi_{\min}(A)$. Therefore, by the Self-correctability Criterion, one has $|\delta(S, J)|=1$. Thus, $d(A)\leqq n(A)+k$ and the first assertion of Corollary 1 follows.

In order to prove the second assertion it will suffice to note that $d(A)=n(A)$ holds for strongly connected definite automata.

Remark 9. The first assertion of Corollary 1 is well-known (e.g. see V. I. Levenshtejn [5, Lemma 11]). In [3] M. Ito and J. Duske obtained the estimation: $|X|^{d(A)}\geqq|S|$.

Although we only dealt with finite automata in this section, some results are valid for arbitrary automata. First, one easily sees that the validity of the Correctability (Self-correctability) Criterion does not depend on the cardinality of the state set and of the input alphabet. One can also prove Decomposition Theorems for arbitrary automata.

A word should be said here about the structure of semigroups of correctable automata. Recall that the semigroup S_A of A is the factor semigroup $X^*/\equiv$ where $J_1\equiv J_2$ iff $\delta(s, J_1)=\delta(s, J_2)$ for all states $s\in S$. It is easy to see that the set of all correcting words forms an ideal of S_A. (Here and elsewhere we do not distinguish between semigroup's elements $J\in S_A$ and corresponding words $J\in X^*$.) If A is finite, then S_A is a finite semigroup, therefore there exists the kernel Ker (S_A) of S_A. One can show that $J\in$Ker (S_A) iff 1) J is a correcting word; and 2) J satifies conditions 1—2 of Theorem 4. Note that conditions 1—2 of Theorem 4 actually means that the set $\{J, J^2, J^3, \ldots\}$ (where $J^2=JJ$, $J^3=JJJ$, ...) forms a subgroup of S_A. Recall that the kernel of an arbitrary compact (in particular, finite) semigroup can be written as a union of pairwise disjoint maximal isomorphic groups: Ker$=\bigcup_\alpha G_\alpha$. The groups G_α are called the group-components of the kernel. If A is a correctable automaton, then each group-component G_α is cyclic and the period of G_α equals $p(A)$. Moreover, $S_A\cdot G_\alpha=G_\alpha$ for each α. One easily see that the semigroups of self-correctable automata possess the following additional property: the equality $S_A\cdot G=G$ holds for any maximal subgroup $G\subseteqq S_A$. In other words, the group-components of S_A (where A is a correctable automaton) are "generalized right zeros" of S_A. If A is self-correctable, then every maximal subgroup of S_A is a "generalized right zero".

Input-induced errors

Here we suggest an equivalent form of A. Ádám's theorem for automata. Let us consider the input-induced errors. Recall that an error (s, t) of storing state t instead of state s is said to be input-induced iff there exist a state v and two words J_1, J_2 of equal length such that $\delta(v, J_1)=s$ and $\delta(v, J_2)=t$. The partition (relation) π is said to be corresponding to the input-induced errors iff π is the smallest (i.e. most refined) partition such that for any input-induced error (s, t) we have $(s, t)\in\pi$. All these concepts were introduced by J. Hartmanis and R. E. Stearns in [2].

Thes next proposition actually was a base of our consideration in the previous section of the paper.

Proposition 3. Let A be an arbitrary (possibly infinite) automaton. Then the partition π corresponding to the input-induced errors is equal to the minimal autonomous partition $\pi_{\min}(A)$.

Note added in Proof. If a connected graph G has at least one semiwalk with positive net length, p of the (unique) cycle of $G/\pi_{\min}$ is equal to the greatest common divisor of all closed semiwalk net lengths of G (G. S. Bloom and S. A. Burr [7, Theorem 3.2]). Elsewise, $G/\pi_{\min}$ has no cycles and consequently $G/\pi_{\min}$ is a directed path (cf. [7, Theorem 3.3]).

Acknowledgement

I am greatly indebted to Professor A. Ádám for many helpful suggestions and comments.

Резюме

Рассматриваются ориентированные графы $G=(V,E)$, где V — множество вершин и E — множество дуг. Разбиение $\pi=\{B_\alpha\}$ множества вершин графа на непересекающиеся блоки $B_\alpha \subseteq V$ называется автономным, если для каждого блока B_α, содержащего хотя бы одну вершину с ненулевой полустепенью исхода, найдется такой блок B_β, что все вершины, достижимые из B_α за один шаг, лежат в B_β. Минимальное автономное разбиение (м.а.р.) графа — это такое его автономное разбиение, которое является собственным подразбиением любого другого автономного разбиения этого графа. Аналоги м.а.р. хорошо известны — это разбиения состояний марковских цепей и автоматов на циклические классы. В нашей работе изучается строение м.а.р. для различных типов ориентированных графов. Мы привели достаточно подробное описание структуры м.а.р. для графов с конечным числом стоков, графов, не содержащих истоков и стоков, а также для сильно связных графов. Можно показать, что м.а.р. графа с конечным числом стоков можно получить из тривиального разбиения этого графа (т.е. разбиения, каждый блок которого содержит в точности одну вершину) путем применения к нему конечного числа операций транзитивного и автономного замыканий, а именно, достаточно $2\times n+2$ таких операций, где n-число стоков графа. При этом для произвольных графов всегда достаточно счетного числа операций. Количество необходимых операций — важная характеристика м.а.р. и его оценкам собственно и посвящена первая часть настоящей статьи.

С помощью м.а.р. оказалось удобным описывать строение автоматов, устойчивых к индуцированным входными искажениями ошибкам. Этим вопросам посвящена вторая часть статьи, где, в частности, решается задача о декомпозиции таких автоматов.

SCIENTIFIC AND PRODUCTION
ASSOCIATION "PROGRAMMPROM"
KLINSKAYA, 6
MOSCOW, 125414
USSR

References

[1] Ádám, A., On certain partitions of finite directed graphs and of finite automata, *Acta Cybernet.* (Szeged), v. 6, 1984, pp. 331—346.
[2] Hartmanis, J. & Stearns, R. E., Algebraic structure theory of sequential machines, *Prentice-Hall, Englewood Cliffs,* New Jersey, 1966.
[3] Ito, M. & Duske, J., On cofinal and definite automata, *Acta Cybernet.* (Szeged), v. 6, 1983, pp. 181—189.
[4] Клосс, Б. Б., Некоторые свойства помехоустойчивых автоматов, *Кибернетика* (Киев), № 1, 1988, 10—15.
[5] Левенштейн, В. И., О некоторых свойствах кодирования и самонастраивающихся автоматах для декодирования сообщений, Пробл. *Кибернетики* (Москва), вып. II, 1964, с. 63—121.
[6] Winograd, S., Input-error-limiting automata, J. for the ACM, v. 11, 1964, pp. 338—351.
[7] Bloom, G. S., Burr, S. A.: On unavoidable digraphs in orientation of graphs, J. Graph Theory, 11 (1987), 453—462.

(Received Dec. 13, 1987)

An approach to automata schemes synthesis

A. S. PODKOLZIN, Š. M. UŠĆUMLIĆ

The function of a finite automaton can be described on both abstract and structural levels. In the first case, only some interrelation between input and output sequences of an automaton is pointed out without showing its structure (we call such automata — abstract). In the second case, an automaton is represented by a scheme, constructed of a set of "elementary" automata, defining a process of conversion of input sequences into output ones (we call such automata structural) [1]. One of the widely used languages, describing the functioning of finite automata on an abstract level is the language of regular expressions [2, 3], which can define time events, distinguishable by automata. In considering structural automata, we shall limit ourselves to schemes constructed out of the following elements: disjunction, conjunction, negation and delay. The problem of constructing a structural automaton V representing an event defined by a regular expression R (in this paper called the problem of synthesis of an automaton V), can be solved by constructing, in an intermediate stage, a Moore diagram of the automaton V [4]. The number of vertices of this diagram in many cases considerably exceeds both the complexity of the scheme of the automaton V and the length of the regular expression R. Corresponding estimation of the number of vertices depends exponentially on the given parameters. In such cases, regardless of a relatively simple scheme definition of an automaton V and a regular expression R, the given method of synthesis becomes in fact, non- applicable, and so arises the necessity of developing "direct" methods of synthesis of the automaton V, which are not based on the construction of its Moore diagram. A direct method of synthesis of automaton schemes, which is in fact an improvement of the synthesis method from [5], is suggested. (From now on these two methods will be reffered to as S_2 and S_1, respectively).

Let us describe the method S_1. The source information in this method is a regular expression R, obtained by application of operations $\cup$, $\cdot$, and $\prec \succ$ over the alphabet $A=\{a_1, a_2, \ldots, a_m\}$ (see [3]). Symbols of the alphabet A are supposed to be encored by binary strings of length $m'=[\log_2 m]$, so that a symbol a_i is encoded by the string $\tilde{a}_i=\tilde{a}_{i1}, \ldots, a_{im'})$; $a_{ij}\in\{0, 1\}$; $i=1, 2, \ldots, m$; $j=1, \ldots, m'$. The regular event defined by a regular expression R is denoted by $|R|$. Let us also introduce the notation $\|R\|=\{\tilde{a}_{i1}\ldots\tilde{a}_{is}|a_{i1}\ldots a_{is}\in|R|,\ s\geqq 1\}$. The problem of synthesis consists of constructing an automaton scheme Σ with m' inputs $x_1, \ldots, x_{m'}$ and one output y, having $\vee$, $\&$ — and delay elements (with 0 or 1 initial state) and representing the event $\|R\|$ by means of the output symbol 1. It means that the appearance of 1 as an output of

the scheme Σ, at a moment t, is equivalent to the belonging of the word $\tilde{a}_{i_1}\tilde{a}_{i_2}\ldots\tilde{a}_{i_t}$, to the event $\|R\|$, where $\tilde{a}_{i_j}$ is a string arriving at the input of the scheme Σ at a moment j; $j=1, \ldots, t$ (we assume that the first moment of time is assigned number 1). Note that in such an approach the empty word e, which generally speaking could be included in $|R|$, is not taken into account.

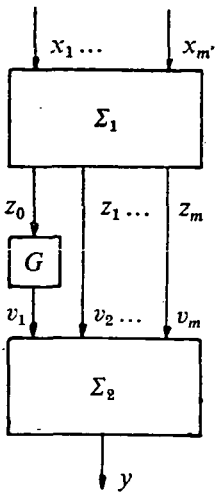

Fig. 1.

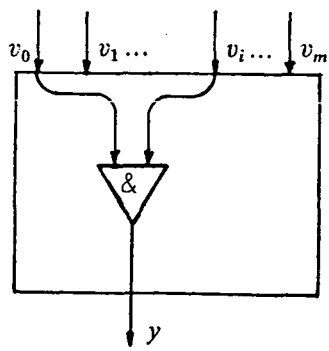

Fig. 2.

According to the method S_1, the scheme Σ is presented in the form given in Fig. 1. The scheme Σ_1 is constructed without using delay elements, its output z_0 is identical to 0, and an output z_i $(i=1, \ldots, m)$ equals 1 if and only if $(x_1, \ldots, x_{m'})=\tilde{a}_i$. Let us denote $\hat{a}_i^{\alpha}=(\alpha, 0, \ldots, 0, 1, 0, \ldots, 0)$; $\alpha\in\{0, 1\}$; $i=1, \ldots, m$. By means of the symbol 1, the scheme Σ_2 represents the event $R=\{\hat{a}_{i_1}^{\alpha_1}\hat{a}_{i_2}^{\alpha_2}\ldots\hat{a}_{i_r}^{\alpha_r}\hat{a}_{i_{r+1}}^{\alpha_{r+1}}\ldots\hat{a}_{i_s}^{\alpha_s}|a_{i_r}\ldots a_{i_s}\in|R|;$ $s\geqq r\}$ and it is defined by induction on the construction of the regular expression R. Let us further denote $\Sigma_2=\Sigma_2(R)$. The initial state of the delay element in Figure 1 equals 1. Construction of the scheme $\Sigma_2(R)$ is implemented in the following way.

1. R has the form a_i. In that case $\Sigma_2(R)$ has the form shown in Fig. 2.
2. R has the form $(R_1\cup R_2)$. In that case $\Sigma_2(R)$ has the form shown in Fig. 3.
3. R has the form $\langle R_1\rangle$. In that case $\Sigma_2(R)$ has the form shown in Fig. 4.
4. R has the form $(R_1\cdot R_2)$. In that case $\Sigma_2(R)$ has the form shown in Fig. 5.

If the regular expression R consists of k_1 occurrences of symbols from the alphabet A, k_2 occurrences of the symbol $\cup$, k_3 occurrences of the symbol $\cdot$ and k_4 ocurrences of the symbol $<\ >$, then the scheme $\Sigma_2(R)$ has the least $k_1+k_2+k_3+2k_4$ elements.

The scheme Σ_1, which is essentially a decoder, can be constructed, for instance, as follows:

1) An input of the negation element is joined with each input $x_1, \ldots, x'_m$. As a result, the values $\bar{x}_1, \ldots, \bar{x}_{m'}$ are computed.

2) Let us consider the oriented tree T shown in Fig. 6. Every vertex w of the tree T, differing from w_0, w_1, w_2 is connected to the conjunction element. Let an edge marked x_i^σ lead from a vertex w' to a vertex w (where $x_i^0=\bar{x}_i$ and $x_i^1=x_i$).
Then the first input of this conjunction element is connected to an input x_i, if $\sigma=1$, and to an output element computing $\bar{x}_i$, if $\sigma=0$. If $w' \notin \{w_1, w_2\}$, then the second input of the conjunction element is connected to an output of the element associated

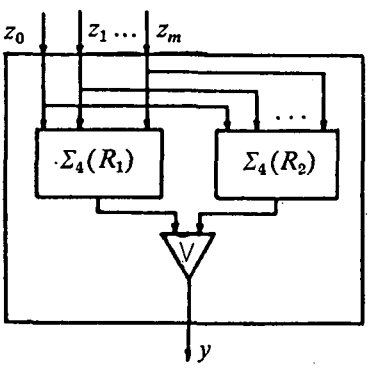

Fig. 3.

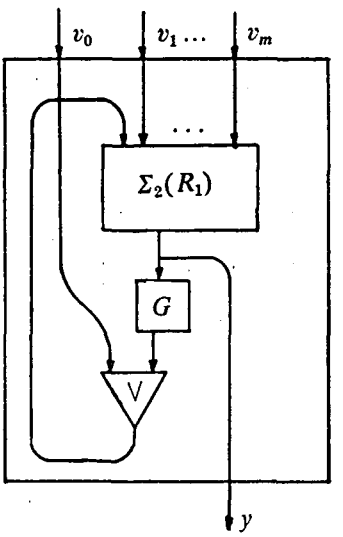

Fig. 4.

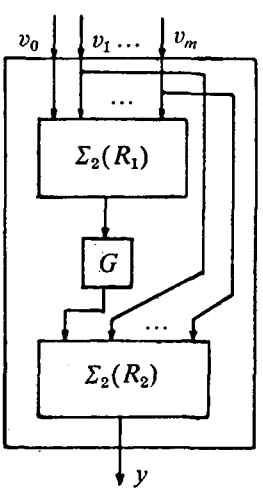

Fig. 5.

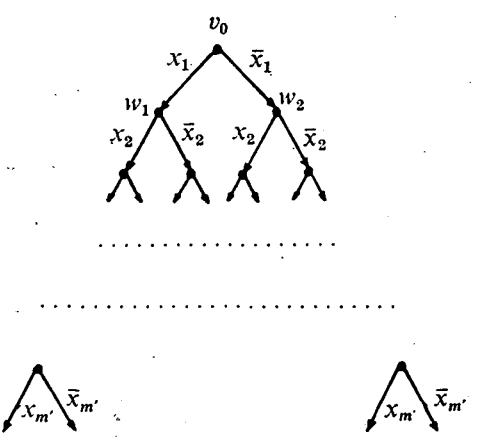

Fig. 6.

with the vertex w'. Otherwise, the second output of the conjunction element is connected to an input x_1, if $w=w_1$, and to an output of element computing $\bar{x}_1$, if $w=w_2$.

3) Conjunction elements, associated with terminating vertices of the tree T, compute all the possible functions of the logic algebra $x_1^{\sigma_1}\ldots x_{m'}^{\sigma_{m'}}$. Let us remove those elements for which $(\sigma_1, \ldots, \sigma_{m'}) \notin \{\tilde{a}_1, \ldots, \tilde{a}_m\}$. The output of the conjunction element, computing the function $x_1^{a_{i1}}\ldots x_{m'}^{a_{im'}}$, is the output z_i of the scheme Σ_1 $(i=1, 2, \ldots, m)$.

4) A conjunction element whose inputs are connected to the input x_1 and to the output of the negation element computing $\bar{x}_1$, is introduced. This conjunction element computes the 0 — function, and its output is the output z_0 of the scheme Σ_1.

It is not difficult to check out that the number of elements of the scheme Σ_1 is $m'+(2^{m'}-4)+m+1$ and that it is not larger then $3m+\log_2 m-2$.

Let us describe the method S_2. Unlike the method S_1, let us assume that the initial regular expression R may contain an empty word symbol e and that $|R| \neq \{e\}$. In the preliminary step, a regular expression R is simplified by the use of the following transformations:

a) $eR' \to R'$; $R' \cdot e \to R'$.

b) $R'' \to e$, if $\|R''\| = \{e\}$ and $R'' \neq e$.

c) Let the expression R include R', which is of the form $\langle R_1 \cup \ldots \cup R_n \rangle$; $n \geqq 1$. If any R_i equals e, then expression $\langle R_1 \cup \ldots \cup R_{i-1} \cup R_{i+1} \cup \ldots \cup R_n \rangle$ is substituted for the expression R'. If an R_i has the form $\langle R_i' \rangle$, then R' is replaced by $\langle R_1 \cup \ldots \cup R_{i-1} \cup R_i' \cup R_{i+1} \cup \ldots \cup R_n \rangle$. If any R_i has the form $R_i' R_i''$ and $e \notin |R_i|$, then $\langle R_1 \cup \ldots \cup R_{i-1} \cup R_i' \cup R_i'' \cup R_{i+1} \cup \ldots \cup R_n \rangle$ is substituted for R'.

d) $R_1 \cdot R_2 \cup R_1 \cdot R_3 \to R_1 \cdot (R_2 \cup R_3)$; $R_2 \cdot R_1 \cup R_3 \cdot R_1 \to (R_2 \cup R_3) \cdot R_1$ (In order to apply this transformation, it is possible to preliminary reorganize disjunctive elements of the expression).

Let us denote using $\tilde{R}$, the result of the described process of simplification of the expression R'. It is not difficult to notice that for any subexpression having the form $\langle Q \rangle$ in $\tilde{R}$ we have $e \notin |Q|$.

A scheme Σ, representing an event $|\tilde{R}|$ is constructed as shown in Fig. 7. where Σ_1 is a scheme constructed in the description of the method S_1 and Σ_3 is a scheme

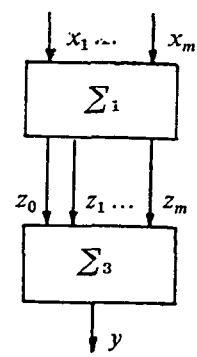

Fig. 7.

representing, using the symbol 1, the event

$$B(\tilde{R}) = \hat{a}_{i1}^{\alpha_1}\hat{a}_{i2}^{\alpha_2}\dots\hat{a}_{is}^{\alpha_s}|a_{i1}\dots a_{is}\in|\tilde{R}|,\ s \geqq 1\}\cup C(\tilde{R}),$$

$$C(\tilde{R}) = \hat{a}_{i1}^{\alpha_1}\hat{a}_{i2}^{\alpha_2}\dots\hat{a}_{ir}^{\alpha_r}\hat{a}_{ir+1}^{\alpha_{r+1}}\dots\hat{a}_{is}^{\alpha_s}|\hat{a}_{i2}\dots\hat{a}_{is}\in|\tilde{R}|;\ s \geqq r\}.$$

To construct the scheme Σ_3, let us define, by induction over the construction of the regular expression $\tilde{R}$, an axilliary scheme $\Sigma_4(\tilde{R})$ representing, by the use of 1, the same event $B(\tilde{R})$ as the scheme Σ_3:

1) $\tilde{R}$ has the form e. Then the scheme $\Sigma_4(\tilde{R})$ has the form given in Fig. 8.

2) $\tilde{R}$ has the form a_i; $i\in\{1, \dots, m\}$. Then $\Sigma_4(\tilde{R})$ has the form given in Fig. 9, where the initial state of the dealy element equals 1.

3) $\tilde{R}$ has the form $(R_1\cup R_2)$. Then $\Sigma_4(\tilde{R})$ has the form given in Fig. 10.

4) $\tilde{R}$ has the form $\langle R_1\rangle$. Then $\Sigma_4(\tilde{R})$ has the form given in Fig. 11.

5) $\tilde{R}$ has the form $R_1\cdot R_2$. Then, in case that $e\in|R_1|$, the scheme $\Sigma_4(\tilde{R})$ has the form given in Fig. 12 and in case that $e\notin|R_1|$, it is derived from the scheme in Fig. 12

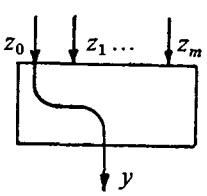

Fig. 8.

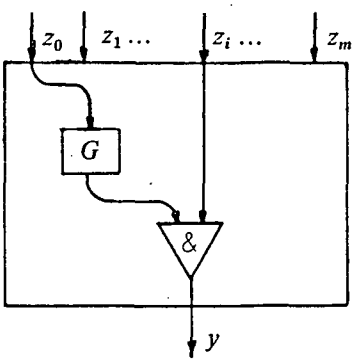

Fig. 9.

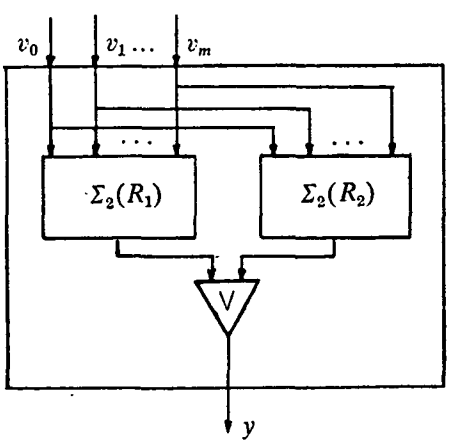

Fig. 10.

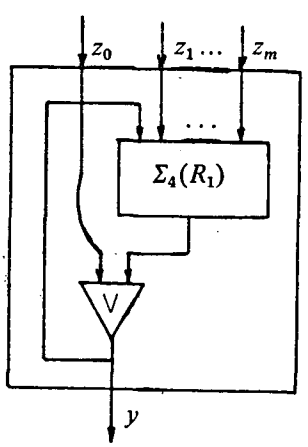

Fig. 11.

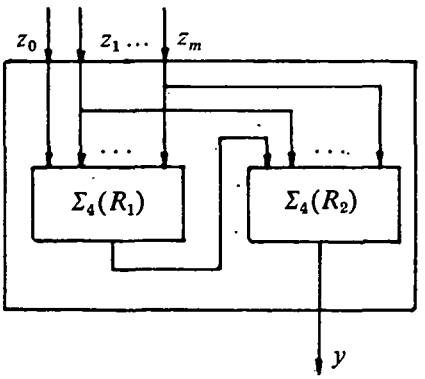

Fig. 12.

by substituting all the 1 — initial states of dealy elements in block $\Sigma_4'(R_2)$ for 0 — initial state. It is not difficult to see that the resulting block Σ (R_2) represents, by means of 1, an event $C(R_2)$.

The scheme Σ_3 derives from the scheme $\Sigma_4(\tilde{R})$ by means of the following transformations:

a) All the delay elements, having the same initial state, inputs of which are associated with an output of the same element or with the same input z_0 of the scheme $\Sigma_4(\tilde{R})$ become identical.

b) All conjunction elements whose inputs are connected to the same output of a delay element and to an input Z_i, become identical.

c) If one of the inputs of the disjunction element E turns out to be associated with the input z_0 of the scheme $\Sigma_4(\tilde{R})$, then the second input of E is identified with the output of this element, and the element itself is removed. If no input of an element of the scheme Σ_3 has been associated with the output z_0 of the scheme Σ_1, then the conjunction element whose output is z_0, is removed.

Let us denote the results of the application of the synthesis methods S_1 and S_2 to a regular expression R by $S_1(R)$ and $S_2(R)$, respectively. We call the number of elements of the scheme Σ the complexity of that scheme and denote it by $L(\Sigma)$. In order to compare the complexity of the schemes $S_1(R)$ and $S_2(R)$, let us define a few auxiliary notions. (Such comparison is possible only for regular expressions R, not containing the symbol e.)

An occurrence an expression of the form $\langle 0\rangle$ in a regular expression $\tilde{R}$. Let a regular expression $R_1 \cup \ldots \cup R_s$, $s \geqq 2$, occur in $\tilde{R}$, where each R_i has the form $\langle Q\rangle$ or $\langle Q\rangle R_i'$. Then an occurrence of $\langle Q\rangle$ in $\tilde{R}$ we call a disjunctive occurrence of iteration. All other occurrences of iterations in $\tilde{R}$ we call no-disjunctive.

Theorem 1. If a regular expression R does not contain the symbol e, then the following inequality holds: $L(S_2(R)) \leqq L(S_1(R)) - N$, where N is the number of nondisjunctive occurrences of iteration in a regular expression $\tilde{R}$.

Proof. If a regular expression R does contain the e symbol, then the transition to expression $\tilde{R}$ is performed only by means of the transitions given in (c) (see above).

The number of iterations doesnot increase, and total number of operations $\cup$ remains the same. Therefore, $L(S_1(\tilde{R}))\leqq L(S_1(\tilde{R}))$. Let us denote by k_1 — the number of occurrences of symbols from the alphabet A in a regular expression R, by k_2 — the number of occurrences of the symbol $\cup$, by k_3 — the number of the occurrences of the symbol $\cdot$, and by k_4 — the number of the occurrences of the symbol $< >$. We have

$$L(S_1(\tilde{R})) \geqq L(\Sigma_1)+k_1+k_2+k_3+2k_4+1;$$

$$L(S_2(R)) \leqq L(\Sigma_1)+2k_1+k_2+k_4-k_1';$$

where k_1' is the number of delay elements removed from the scheme $\Sigma_4(\tilde{R})$ during its transformation into the scheme Σ_3. It is not difficult to notice that the number of occurrences of letters in the regular expression is one more than the number of occurrences of binary operations, i.e., $k_1=k_2+k_3+1$. Therefore $L(S_1(\tilde{R})\geqq L(\Sigma_1)+2k_1+$ $+2k_4$. Thus, for the proof of the theorem it is enough to prove the inequality $L(\Sigma_1)+2k_1+k_2+k_2+k_4-k_1'\leqq L(\Sigma_1)+2k_1+2k_4-N$, or after reduction: $k_2-k_1'\leqq$ $\leqq k_4-N$. Let Π be an occurrence of the expression $\tilde{R}$ having the form $R_1\cup R_2$; $i\in\{1, 2\}$, where each R_i has the form $R_{i1}\cdot R_{i2}\ldots R_{is}$; $s\geqq 1$; R_{i1} is not of the form of $R'\cdot R''$. In that case we say that the occurrence of the expression R_{i1} in $\tilde{R}$ is subordinated to the occurrence of Π. Let us connect, with a regular expression $\tilde{R}$, an oriented graph G, whose vertices are the occurrence of the expression $R_1\cup R_2$ in R and also the occurrences subordinated to them. An edge leads from a vertex v to a vertex w in the graph G, if and only if the occurrence w is subordinated to the occurrence v. It is not difficult to notice that every occurrence in a regular expression $\tilde{R}$ is subordinated to not more than one occurrence. Therefore, the graph G represents the union of a finite number of oriented trees $G_1, \ldots, G_r$. Let q_i denote the number of nonterminating vertices of the tree G_i; $i=1, \ldots, r$. Evidently, $\sum_{i=1}^{r} q_i=k_2$. As from each nonterminating vertex of the tree G_i exactly two edges leave, the number of terminating vertices of the tree G_i equals q_i+1. Let q_i' be the number of terminating vertices of the tree G_i representing the occurrences of letters from the alphabet A. Then q_i+1-q_i' is the number of terminating vertices of the tree G_i representing occurrences of iterations and all these occurrences are disjunctive. It is not difficult to see that delay elements corresponding to the occurrences of letters from the alphabet A in $\tilde{R}$, which are terminating vertices of the tree G_i, become identical through the transformation of the scheme $\Sigma_4(\tilde{R})$ into the scheme Σ_3. Therefore, denoting $q_i''=q_i'$ when $q'=0$, and $q_i''=q_i'-1$ when $q_i>0$, we have: $\sum_{i=1}^{r} q_i''\leqq q_i'$. Consequently, $q_2-k_1'\leqq$ $\leqq \sum_{i=1}^{r}(q_i-q_i'')$. On the other hand, k_4-N is equal to the number of occurrences of disjunctive iterations in $\tilde{R}$, and consequently, $\sum_{i=1}^{r}(q_i+1-q_i')\leqq k_4-N$. When $q_i'=0$ we have: $q_i-q_i''=q_i\leqq q_i+1=q_i+1-q_i'$; however, if $q_i>0$ then $q_i-q_i'=$ $=q_i-q_i'+1$. This implies the inequality $\sum_{i=1}^{r}(q_i-q_i'')\leqq \sum_{i=1}^{r}(q_i+1-q_i')$. The theorem is proved.

Construction of the scheme Σ_3 by the method S_2 is realized by induction on the structure of a regular expression $\tilde{R}$, with further transformations of the scheme. Let us give a more convenient constructing procedure of the scheme Σ_3, based on a preliminary marking of a regular expression $\tilde{R}$. The marking gives the possibility to follow directly a series of removals of elements from the scheme $\Sigma_4(\tilde{R})$ during its transformation into the scheme Σ_3, as well as the joining together of the elements of that scheme.

Let us define a series of auxiliary notions connected to a regular expression $\tilde{R}$. Occurrences of letters Π_1 and Π_2 from the alphabet A in the regular expression $\tilde{R}$ are called similar if there exist such sequences of occurrences $\Pi_0, \Pi_1, \ldots, \Pi_s=\Pi$ and $\Pi_0, \Pi_1', \ldots, \Pi_r'=\Pi$ in $\tilde{R}$, in which every next occurrence is subordinated to the previous one. Similar occurrences of the same letter in $\tilde{R}$ we call adjacent. Therefore, every class of similar occurrences of letters in $\tilde{R}$ is devided into a number of subclasses of adjacent occurrences.

Let Π be an occurrence of a regular expression R' in the regular expression $\tilde{R}$. We shall now define occurrences in $\tilde{R}$, referred to as the basis and the predecessor of the occurrence Π:

1. If $R'=e$, then the basis of Π is equal to the predecessor of Π.

2. If $R'=a_i$, $i\in\{1, \ldots, m\}$, then the basis of Π represents a distinguished element of the class of occurrences of the letter a_i, occurrences being adjacent with Π (all elements of this class have the same basis). The occurrence of a letter is called basic if this occurrence is included in the basis.

3. If $R'=R_1 \cup R_2$ and if the predecessor of Π is defined or both expressions R_1 and R_2 differ from e, then the basis of the occurrence Π is Π. If the predecessor of the occurrence Π is undefined and $R_{i1}=e$; $\{i_1, i_2\}=\{1, 2\}$, then the basis of the occurrence Π is equal to the basis of the occurrence of R_{i2} in $\tilde{R}$. In all the enumerated cases the predecessors of the occurrences of R_1 and R_2 in $\tilde{R}$ are equal to the predecessor of the occurrence Π.

4. If $R'=\langle R_1\rangle$, and the predecessor of the occurrence Π is defined, then the basis of the occurrence Π equals Π; otherwise it is equal to the basis of the occurrence of R_1 in $\tilde{R}$. If the predecessor of the occurrence Π is defined, then the predecessor of the occurrence of in $\tilde{R}$ is Π; otherwise it is equal to the basis of the occurrence of R_1 in $\tilde{R}$.

5. If $R'=R_1 \cdot R_2$, then the basis of the occurrence Π is equal to the basis of the occurrence of R_2 in $\tilde{R}$. The predecessor of the occurrence R_2 in $\tilde{R}$ is equal to the basis of the occurrence R_1 in $\tilde{R}$; the predecessor of the occurrence R_1 in $\tilde{R}$ is equal to the predecessor of the occurrence Π.

6. If $R'=\tilde{R}$, then the predecessor of the occurrence Π is undefined (at the some time all the predecessors as well as basis of the occurrences in $\tilde{R}$, which, according to p. 1—5 are equal to the predecessor of the occurrence Π, are undefined).

It is not difficult to check out that, in accordance with p. 1—6, for any occurrence of a regular expression in $\tilde{R}$, it is possible to unambigously find (in a finite number of steps), the basis and the predecessor of that occurrence, or their absence can be confirmed.

Next, let us define initial occurrences in the expression $\tilde{R}$:

1. An occurrence of the expression $\tilde{R}$ in itself is initial.

2. If an occurrence of the expression $\langle R_1\rangle$ in $\tilde{R}$ is initial, then an occurrence of the expression R_1 in $\tilde{R}$ is also initial.

3. If an occurrence of the expression $R_1 \cup R_2$ in $\tilde{R}$ is initial, then the occurrences of the expressions R_1, R_2 in $\tilde{R}$ are initial.

4. If an occurrence of the expression $R_1 \cdot R_2$ in $\tilde{R}$ is initial, then the occurrence of the expression R_1 in $\tilde{R}$ is initial. If at the same time, $e \in |R_1|$, then the occurrence of the expression R_2 in $\tilde{R}$ is initial.

Let us describe, using the notions given above, the process of construction of the scheme Σ_3' obtained from $\Sigma_4(\tilde{R})$ by applying a portion of the transformations a)—c). The scheme Σ_3 will be obtained from Σ_4 by applying the unused of the transformations a) and b). The scheme Σ_3 is constructed in the following way:

1) For each class $\varkappa$ of similar occurrences of letters in $\tilde{R}$, a delay element, denoted by $G(\varkappa)$, is introduced. If all the occurrences from $\varkappa$ are initial then the initial state of this element equals 1, otherwise it is euqal to 0.

2) For each basis occurrence Π of a letter in an expression $\tilde{R}$, an element of conjunction, denoted by $E(\Pi)$, is introduced.

3) For each occurrence Π in the expression $\tilde{R}$ of the expression $R_1 \cup R_2$, such that $R_1 \neq e$ and $R_2 \neq e$ or the predecessor of Π is defined, an element of disjunction, denoted by $E(\Pi)$, is introduced.

4) For each occurrence Π having a predecessor, in expression $\tilde{R}$ of the expression $\langle R_1 \rangle$, an element of disjunction, denoted by $E(\Pi)$, is introduced.

5) The input of the element $G(\varkappa)$ is associated with the output of the element $E(\Pi')$ where Π' is a predecessor of an arbitrary occurence Π belonging to the class $\varkappa$. If the occurrence from $\varkappa$ do not have predecessors, then the input of the element $G(\Pi)$ is associated with the input z_0.

6) If Π is a basic occurrence of the letter a_i, which belongs to the class $\varkappa$ of similar occurrences of letters, then one of the inputs of the element $E(\Pi)$ is joined to the output of the element $G(\varkappa)$ and the other is joined to the input z_i.

7) If Π is an occurrence of the expression in $\tilde{R}$ and the element $E(\Pi)$ is defined, then the inputs of that element are connected with the outputs of the elements $E(\Pi_1)$ and $E(\Pi_2)$, where Π_1 and Π_2 are the basis of the occurrences of R_1 and R_2 in $\tilde{R}$.

8) If Π is an occurrence of the expression $\langle R_1 \rangle$ in $\tilde{R}$ and the element $E(\Pi)$ is defined, then the inputs of this element are connected with the inputs of the elements $E(\Pi_1)$ and $E(\Pi_2)$, where Π_1 is the predecessor of occurrence of Π and Π_2 is the basis of occurrence of R_1 in $\tilde{R}$.

9) The output of the scheme Σ_3' is the output of the element $E(\Pi)$, where Π is the basis of occurrence of $\tilde{R}$ in $\tilde{R}$ (this basis is defined as $|\tilde{R}| = \{e\}$).

Let us illustrate the method S_2 for a regular expression $R = \langle\langle a \rangle b \cup c \langle a \rangle\rangle$ in a three — letter alphabet $A = \{a, b, c\}$. Let us encode the symbols a, b, c, by binary strings 00, 01, 10 respectively. In this case the decoder Σ_1 has the from given in Fig. 13. There are no similar occurrences of letters in the expression R and therefore, delay and conjunctinn elements correspond to each occurrence of a letter. The form of the scheme Σ_3' is given in Fig. 14. Scheme Σ_3 is obtained from the scheme Σ_3' by the unification of delay element corresponding to the initial occurrences of letter a and letter b. Since no element in Σ_3 is connected to the output z_0 of Σ_1, the corresponding conjunction element in Σ_1 is to be removed. The form of the scheme Σ is given in Fig. 15.

Let us note by $L(\tilde{R})$ the least complexity of the schemes, representing, by the symbol 1, an event $|R|$ defined by a regular expression R. The complexity of the regular expression R is the number of occurrences of the letters and operation symbols

$\cup, \ldots, \langle\,\rangle$ within this expression. Let us consider the Shenon function $L(m,n) = \max_{R \in R_{m,n}} \tilde{L}(R)$, where $R_{m,n}$ is the class of all regular expressions in an m — letter alphabet A with complexity not greater than n. According to the synthesis method S_2, an estimation $\tilde{L}(m,n) \leqq 3m + \log_2 m - 2 + 2n$ holds. In the paper [6], there is an example of a regular expression — with complexity n in a two — letter alphabet, for

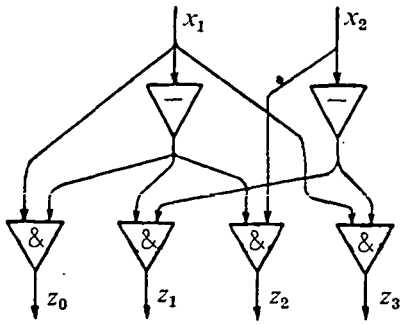

Fig. 13.

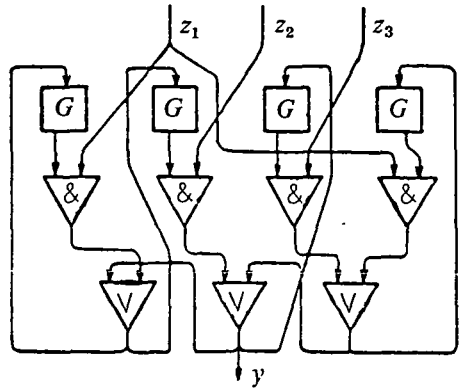

Fig. 14.

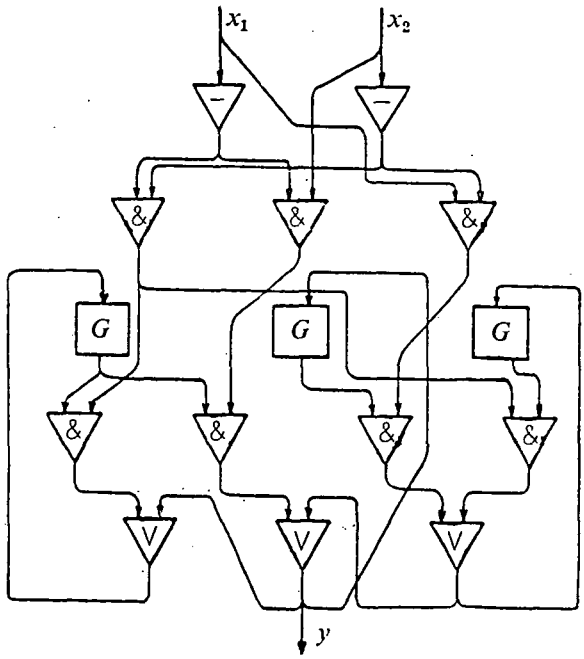

Fig. 15.

which the minimal number of delay elements in the corresponding scheme is asymptotically not less than $\frac{n}{5}$. It turns out that, for a fixed $m \geqq 2$ and $n \to \infty$, the following asymptotical inequalities hold: $\frac{n}{5} \lesssim \tilde{L}(m, n) \leqq 2$, i.e., the order of the entity $\tilde{L}(m, n)$ equals n.

As a conclusion, let us describe one of the methods of simplifying the scheme $\Sigma_4(\tilde{R})$, used in the process of realization of the method S_2 without marking the expression $\tilde{R}$. Let $\Pi_1, \ldots, \Pi_k$ be non — initial occurrences of the same regular expression R' in $\tilde{R}$. We call the occurrences $\Pi_1, \ldots, \Pi_k$ alternative, if for any letter p in the alphabet A, the set of pre-main positions of the expression $\tilde{R}$, p — following its starting positions (see [3]) has an empty intersection with the set of pre-main positions of not more then one of the occurrences $\Pi_1, \ldots, \Pi_k$. For example, the first and the second occurrences of the expression $(b\langle a\rangle \cup c)$ in the regular expression $a(b\langle a\rangle \cup c) \cup \cup\langle b\langle a\rangle\rangle \cup c\rangle$ are alternative.

Let $\Sigma_4(\tilde{R})$ be a scheme constructed by using the method S_2 for a regular expression $\tilde{R}$ and let $\Pi_1, \ldots, \Pi$ (be the alternative occurrences of a regular expression R' in $\tilde{R}$. Let us determine, in the scheme $\Sigma_4(\tilde{R})$, the blocks $B_1, \ldots, B_k$ corresponding to the occurrences $\Pi_1, \ldots, \Pi_k$. Each such block represents the scheme $\Sigma_4'(R')$ and at any instant of time t, all the blocks $B_1, \ldots, B_k$, except possibly one, have a zero state of delay elements. This enables us to use, in the scheme $\Sigma_4(\tilde{R})$, only one block $\Sigma_4(\tilde{R}')$ instead of k samples of such blocks. This block switches its input z_0 and output according to the positions of blocks B_i; $i=1, \ldots, k$. This switching will be realized using the scheme W given in Fig. 16. If $e \notin |R'|$ then the first input of the conjunction

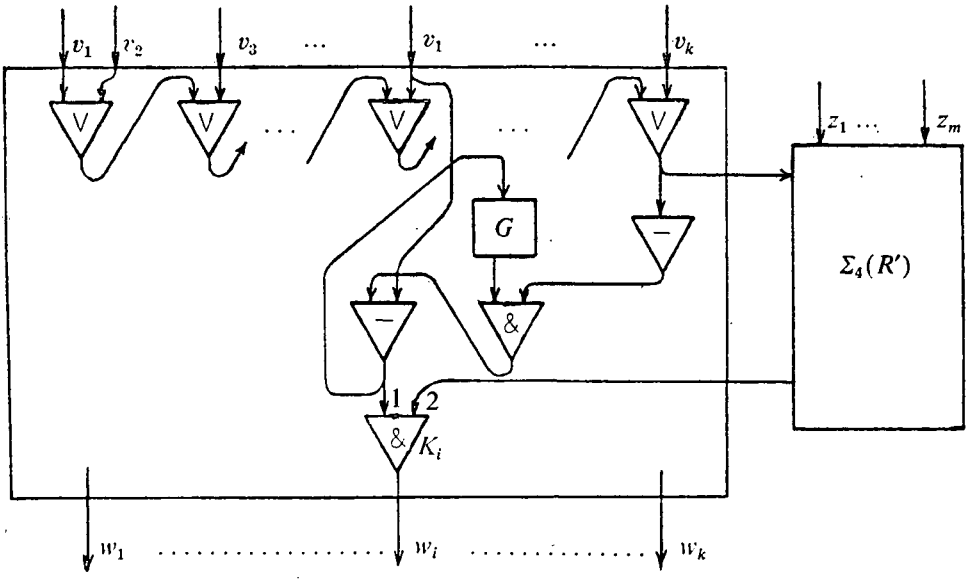

Fig. 16.

element K_i is joined to the output of the element z_i; $i=1,\ldots,k$. The input v_i of this scheme is joined either to the same element, or the input of $\Sigma_4(\tilde{R})$, as well as the input z_0 of the block B_i. The input w_i is joined to the inputs of the same elements or the output of the scheme $\Sigma_4(\tilde{R})$, as well as the output of the block B_i; $i=1,\ldots,k$. At the initial instant of time, states of the dalay element $z_1,\ldots,z_k$ of the scheme W equal 0. When 1 arrives to the input v_i, the block $\Sigma_4'(R')$ begins to participate in the functioning of the scheme $\Sigma_4(\tilde{R})$ as a block B_i, while the delay element z_i turns into state 1 and others into state 0. The complexity of the scheme W is $5k+L(\Sigma_4'(R'))$ and therefore, applying the described process of exchange of block $B_1,\ldots,B_k$ to the scheme W is useful only in case that $5K+L(\Sigma_4'(R'))\leqq k\cdot L(\Sigma_4'(R'))$, i.e., when $L(\Sigma_4'(R'))\geqq\frac{5k}{k-1}$.

АЛЕКСАНДАР СЕРГЕЕВИЧ ПОДКОЛЗИН
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. М. В. ЛОМОНОСОВА
МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
СССР, 117234 МОСКВА, В-234, ЛЕНИНСКИЕ ГОРЫ, МГУ

ŠĆEPAN UŠĆUMLIĆ
UNIVERSITET U BEOGRADU
TEHNOLOŠKO-METALURŠKI FAKULTET
KATEDRA ZA MATEMATIČKE NAUKE
JUGOSLAVIJA, 11000 BEOGRAD, KARNEDŽIJEVA 4

Bibliography

[1] Кудряьцев, В. Б., Алёшин, С. В., Подколзин, А. С.: Элементи теории автоматов, Москва, *Изд-во Моск. университета*, 1978.
[2] Клини, С. К.: Представление событий в нервных сетях и конечных автоматах, *В книге «Автоматы»*, Москва, ИЛ, 1956, стр. 15—67.
[3] Кудрявцев, В. Б., Подколзин, А. С., Ушчумлйч, Ш. М.: Введение в теорию абстрактных автоматов, Москва, *Изд-во Моск. университета*, 1985.
[4] Глушков, В. М.: Синтез цифровых автоматов, Москва, *Физмат.*, 1962.
[5] Копи, И. М., Элгот, К. С., Райт, Д. Б.: Реализация событий логическими сетями. *В книге Кибернетический сборник*, Вып. 3, М. ИЛ, 1961, 147—166.
[6] Гринберг, В. С.: Детерминизация систем графов и синтез конечных автоматов, *Сиб. матем. журнал*, Том 7, №. 6, 1966, 1260—1267.

(Received Dec. 8, 1987)

The finite source queueing model for multiprogrammed computer systems with different CPU times and different I/O times

BRIAN D. BUNDAY and ESMAILE KHORRAM

Abstract

This paper discusses the finite source queueing model as it applies to a multiprogrammed computer system. The system processes N jobs using r Central Processing Units (CPU's) where $r<N$. The jobs emanate from peripheral devices, terminals, card readers etc. (I/O devices) at which it is assumed they suffer no delay.

If a CPU is available when a job requires service it is given this service. Otherwise a queue of jobs is formed. In the situation where there are more than r jobs requiring service, it is assumed that r randomly selected jobs are assigned to each of the r CPU's. It is assumed that the service time of job i has a negative exponential distribution with mean $1/\mu_i$. After service, job i returns to I/O devices for a random time before again calling for CPU service. This time is assumed to have a general distribution with mean $1/\lambda_i$.

A closed form solution for the steady-state probabilities that a particular set of jobs is at I/O processes is obtained. It is shown that the steady-state solution depends on the distribution of time at I/O devices only through the value $1/\lambda_i$. It is also shown how other important measures such as CPU utilisation, as well as waiting times and response times for the jobs, can be computed from this solution.

1. Introduction

A number of authors have applied the methods of queueing theory to the study of multiprogrammed computer systems. Following Sztrik [11] we can model such systems as follows. We suppose that there are N jobs in the system, each one emanating from a terminal at which it suffers no delay and to which it returns following CPU processing. There are $r(<N)$ CPU's in the system. If a CPU is available an arriving job (program) is immediately served by one of the available CPU's. Otherwise a queue of jobs is formed. The jobs would normally be served on a FIFO (first-in, first-out) basis. For job i we assume that its service time is exponentially distributed with mean $1/\mu_i$. We also suppose that the time job i spends at the peripheral devices (I/O operations) is a random variable with distribution function $F_i(x)$ or more conveniently survivor function $G_i(x)=1-F_i(x)$. These times are independent of each other and are different for the different jobs.

The queueing model just described was first used in the context of the "machine interference problem" by Ashcroft [1] who studied the $M/G/1$ case by way of the duration of the busy period of the operative (the CPU). Using the birth-death equa-

tions Benson and Cox [3] obtained a solution to the $M/M/1$ case and extended it to the $M/M/r$ case for which Peck and Hazelwood [9] computed extensive tables for work study applications. An important advance was made by Bunday and Scraton [4] who showed that the solution to the $G/M/r$ homogenenous case was the same as the $M/M/r$ solution.

There is a considerable literature showing applications of this and similar models to the computer systems situation. Early contributions were made by Gaver [6] and Avi-Itzhak and Heyman [2] while more recently we have the papers by Cohen [5], Schatte [10], Kameda [7] and Sztrik [11, 12]. The book by Kleinrock [8] contains an extensive bibliography as well as a discussion of other models.

The present paper extends the work of Sztrik and presents a closed form solution of the $\vec{G}/\vec{M}/r$ case in the steady-state situation for a queue discipline in which jobs are randomly allocated to CPU's whenever a new job calls for service or the service of a job is completed. From this it is easy to compute such quantities as the CPU utilisation and the expected waiting times and response times of the jobs. It is shown that these quantities depend on the distribution of the times spent at the I/O processes only through the means of these distributions.

2. The steady-state equations for the model

We consider a set of N jobs in a system with r CPU's. Service times for each job are assumed to have a negative exponential distribution with mean $1/\mu_i$ for job i. The times spent at I/O processes for each job are independently distributed.

Let $G_i(t)$ denote the probability that if job i arrives at I/O processes at time zero then it is still there at time t later. Thus

$$G_i(0) = 1 \quad \text{and} \quad G_i(\infty) = 0 \quad \text{for all } i.$$

Further if job i is at I/O processes at time t the probability that it will call for CPU service in the interval $(t, t+\delta t)$ is

$$-G_i'(t)\,\delta t/G_i(t) \quad \text{to first order in } \delta t. \tag{2.1}$$

The mean time spent at I/O by job i will be

$$\frac{1}{\lambda_i} = \int_0^\infty t[-G_i'(t)]\,dt = \int_0^\infty G_i(t)\,dt \tag{2.2}$$

Let $Q_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2} \ldots, t_{i_n}; \tau)\,dt_{i_1} dt_{i_2}, \ldots, dt_{i_n}$ be the probability that at time τ a particular set $i_1, i_2, \ldots, i_n$ of the N jobs are at I/O, one of them for a time in $(t_{i_1}, t_{i_1}+dt_{i_1})$, etc,, ..., another for a time in $(t_{i_n}, t_{i_n}+dt_{i_n})$, and the other jobs require CPU service. In the case of negative exponential service, $n, t_{i_1}, t_{i_2}, \ldots, t_{i_n}$, and τ provide an adequate description of the system. We do not need to specify the state of each service at time τ since this will not influence the future behaviour of the system. Indeed we need not even specify which particular jobs are being serviced since the residual service time has the same distribution whether or not the service has been started.

We consider the transitions that may occur in $(\tau, \tau+\delta\tau)$ working always to first order in $\delta\tau$.

$$Q_{i_1 i_2 \ldots i_N}(t_{i_1}+\delta\tau, \ldots, t_{i_N}+\delta\tau;\ \tau+\delta\tau) =$$

$$= Q_{i_1, i_2 \ldots i_N}(t_{i_1}, t_{i_2}, \ldots, t_{i_N};\ \tau)\left[1-\delta\tau \sum_{s=1}^{N}\{-G'_{i_s}(t_{i_s})/G_{i_s}(t_{i_s})\}\right]. \tag{2.3}$$

It is convenient to denote $\{i_1 i_2 \ldots i_n\}$ the set of jobs at I/O by A_n while $B_n = A_n^c$ denotes the set of jobs calling for CPU service.

$$Q_{i_1, i_2 \ldots i_n}(t_{i_1}+\delta\tau, \ldots, t_{i_n}+\delta\tau;\ \tau+\delta\tau) =$$

$$= Q_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n};\ \tau)\left[1-\delta\tau \sum_{s=1}^{n}\{-G'_{i_s}(t_{i_s})/G_{i_s}(t_{i_s})\} - \delta\tau \sum_{j\in B_n}\mu_j\right]+$$

$$+\delta\tau \sum_{j\in B_n}\int_0^{\tau} Q_{i_1 i_2 \ldots i_n, j}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}, t_j;\ \tau)\{-G'_j(t_j)/G_j(t_j)\}\, dt_j \tag{2.4}$$

for all groups $i_1, i_2, \ldots, i_n$ such that $N-r \leqq n \leqq N-1$.

$$Q_{i_1 i_2 \ldots i_n}(t_{i_1}+\delta\tau, \ldots, t_{i_n}+\delta\tau;\ \tau+\delta\tau) =$$

$$= Q_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n};\ \tau)\left[1-\delta\tau \sum_{s=1}^{n}\{-G'_{i_s}(t_{i_s})/G_{i_s}(t_{i_s})\} - \frac{r}{N-n}\delta\tau \sum_{j\in B_n}\mu_j\right]+$$

$$+\delta\tau \sum_{j\in B_n}\int_0^{\tau} Q_{i_1 i_2 \ldots i_n, j}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}, t_j;\ \tau)\{-G'_j(t_j)/G_j(t_j)\}\, dt_j \tag{2.5}$$

for all groups $i_1, i_2, \ldots, i_n$ such that $1 \leqq n \leqq N-r$.

In (2.5) the particular set of r jobs being serviced is equally likely to be any one of the $\binom{N-n}{r}$ sets possible. This is equivalent to assuming that whenever the number of jobs requiring service exceeds r, then we have a SIRO (service in random order) queue discipline. This is a somewhat artificial situation and is certainly different from the more natural FIFO discipline. In the latter case the resulting system of equations has no explicit solution. We shall show that for the queue discipline adopted the equations can be solved. In many cases, provided the inhomogeneity is not excessive, our solution, particularly in respect of the important properties of the system, will be a good approximation to the FIFO case. Its closed and easily computed form is its merit.

$$Q_0(\tau+\delta\tau) = Q_0(\tau)\left[1-\frac{r}{N}\delta\tau \sum_{j=1}^{N}\mu_j\right]+$$

$$+\delta\tau \sum_{j=1}^{N}\int_0^{\tau} Q_j(t_j;\ \tau)\{-G'_j(t_j)/G_j(t_j)\, dt_j. \tag{2.6}$$

If we consider the situation when a service is completed in the interval $\tau, \tau+\delta\tau$

$$Q_{i_1 i_2 \ldots i_n j}(t_{i_1}+\delta\tau, t_{i_2}+\delta\tau, \ldots, t_{i_n}+\delta\tau, 0;\ \tau+\delta\tau)\cdot\delta\tau =$$

$$= \mu_j \delta\tau Q_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n};\ \tau) \tag{2.7}$$

for all $j \in B_n$ and all groups $i_1, \ldots, i_n$ such that $N-r \leqq n \leqq N-1$.

$$Q_{i_1 i_2 \ldots i_n j}(t_{i_1}+\delta\tau, t_{i_2}+\delta\tau, \ldots, t_{i_n}+\delta\tau, 0;\ \tau+\delta\tau)\cdot\delta\tau =$$

$$= \frac{r\delta\tau}{N-n}\mu_j Q_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n};\ \tau) \tag{2.8}$$

for all $j \in B_n$ and all groups $i_1, \ldots, i_n$ such that $0 \leqq n \leqq N-r$.

If we consider the situation as $\tau \to \infty$ so that

$$Q_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n};\ \tau) \to Q_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n})$$

and further write

$$Q_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}) = G_{i_1}(t_{i_1}) G_{i_2}(t_{i_2}) \ldots G_{i_n}(t_{i_n}) R_{i_1 i_2 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}) \tag{2.9}$$

then (2.3) to (2.8) take the form

$$\left[\frac{\partial}{\partial t_{i_1}} + \ldots + \frac{\partial}{\partial t_{i_N}}\right] R_{i_1 \ldots i_N}(t_{i_1}, t_{i_2}, \ldots, t_{i_N}) = 0 \tag{2·10}$$

$$\left[\frac{\partial}{\partial t_{i_1}} + \ldots + \frac{\partial}{\partial t_{i_n}}\right] R_{i_1 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}) = -R_{i_1 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}) \sum_{j \in B_n} \mu_j -$$

$$- \sum_{j \in B_n} \int_0^\infty R_{i_1 \ldots i_n j}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}, t_j) G_j'(t_j)\, dt_j \tag{2.11}$$

for all groups $i_1, \ldots, i_n$ such that $N-r \leqq n \leqq N-1$.

$$\left[\frac{\partial}{\partial t_{i_1}} + \ldots + \frac{\partial}{\partial t_{i_n}}\right] R_{i_1 \ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}) = -\frac{r}{N-n} R_{i_1 i_2 \ldots i_n}(t_{i_1}, \ldots, t_{i_n}) \sum_{j \in B_n} \mu_j -$$

$$- \sum_{j \in B_n} \int_0^\infty R_{i_1 \ldots i_n j}(t_{i_1}, \ldots, t_{i_n}, t_j) G_j'(t_j)\, dt_j \tag{2.12}$$

for all groups $i_1, \ldots, i_n$ such that $1 \leqq n \leqq N-r$.

$$0 = R_0 \frac{r}{N} \sum_{j=1}^{N} \mu_j + \sum_{j=1}^{N} \int_0^\infty R_j(t_j) G_j'(t_j)\, dt_j. \tag{2.13}$$

$$R_{i_1 \ldots i_n j}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}, 0) = \mu_j R_{i_1 \ldots i_n}(t_{i_1}, \ldots, t_{i_n}) \tag{2.14}$$

for all $j \in B_n$ and groups $i_1, \ldots, i_n$ such that $N-r \leqq n \leqq N-1$.

$$R_{i_1 \ldots i_n j}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}, 0) = \frac{r}{N-n} \mu_j R_{i_1 \ldots i_n}(t_{i_1}, \ldots, t_{i_n}) \tag{2.15}$$

for all $j \in B_n$ and groups $i_1, \ldots, i_n$ such that $0 \leqq n \leqq N-r$.

It is perhaps worth mentioning that these same equations would result from a second queue discipline which Tomkó refers to as "processor sharing". In this situa-

tion whenever there are more jobs demanding service than CPU's, i.e. $N-n>r$, then all jobs receive service on each CPU in such a way that during a unit time every job receives an amount $1/(N-n)$ CPU service on every CPU. This will approximate the case when all CPU's operate in time sharing. See also Cohen [5].

3. The solution of the steady state equations

The general solution of (2.10) is

$$R_{i_1\ldots i_N}(t_{i_1}, t_{i_2}, \ldots, t_{i_N}) = g(t_{i_1}-t_{i_2}, t_{i_1}-t_{i_3}, \ldots, t_{i_1}-t_{i_N})$$

where g is an arbitrary function.

But $R_{i_1\ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n})$ is a symmetric function for all A_n so that

$$R_{i_1\ldots i_N}(t_{i_1}, t_{i_2}, \ldots, t_{i_N}) = \varkappa \tag{3.2}$$

where $\varkappa$ is a constant is a solution.

From (2.14) and (2.15) we obtain in turn

$$R_{i_1\ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}) = \frac{\varkappa}{\prod_{k=1}^{N-n} \mu_{j_k}} \tag{3.3}$$

where $j_k \in B_n$ and $N-r \leqq n \leqq N-1$.

$$R_{i_1\ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n}) = \frac{(N-n)!\,\varkappa}{r^{N-n-r} r! \prod_{k=1}^{N-n} \mu_{j_k}} \tag{3.4}$$

where $j_k \in B_n$ and $0 \leqq n \leqq N-r$.

These solutions also satisfy (2.11) to (2.13).

Thus the probability that a particular group $i_1, i_2, \ldots, i_n$ of jobs are at I/O and the rest are not is

$$Q_{i_1 i_2\ldots i_n} = \int_0^\infty \cdots \int_0^\infty Q_{i_1 i_2\ldots i_n}(t_{i_1}, t_{i_2}, \ldots, t_{i_n})\, dt_{i_1} \ldots dt_{i_n}$$

so that from (2.2)

$$Q_{i_1\ldots i_n} = \frac{(N-n)!\,\varkappa}{r^{N-n-r} r! \prod_{k=1}^{N-n} \mu_{j_k}} \prod_{j=1}^{n} \lambda_{i_j}^{-1} \tag{3.5}$$

for all groups with $0 \leqq n \leqq N-r$.

$$Q_{i_1\ldots i_n} = \frac{\varkappa \prod_{j=1}^{n} \lambda_{i_j}^{-1}}{\prod_{k=1}^{N-n} \mu_{j_k}} \tag{3.6}$$

for all groups with $N-r \leqq n \leqq N$.

Thus the probability that n jobs are at I/O is

$$q_n = \sum_{\{i_1\ldots i_n\}} \frac{(N-n)!\,\varkappa}{r^{N-n-r} r! \prod_{k=1}^{N-n} \mu_{j_k}} \prod_{j=1}^{n} \lambda_{i_j}^{-1} \tag{3.7}$$

for $0 \leqq n \leqq N-r$.

$$q_n = \sum_{\{i_1 \ldots i_n\}} \frac{\varkappa}{\prod_{k=1}^{N-n} \mu_{j_k}} \prod_{j=1}^{n} \lambda_{i_j}^{-1} \tag{3.8}$$

for $N-r \leqq n \leqq N$.

$\varkappa$ is determined by the condition

$$\sum_{n=0}^{N} q_n = 1. \tag{3.9}$$

4. Some useful measures for the system

The probability that all N jobs are at I/O is

$$q_N = \varkappa / [\prod_{j=1}^{N} \lambda_j]. \tag{4.1}$$

Since if n jobs are at I/O the probability that a particular CPU is servicing is $\frac{N-n}{r}$ if $N-n \leqq r$ or 1 if $N-n > r$ then the proportion of time each CPU is servicing, the CPU utilisation, is given by

$$U = [\sum_{k=1}^{r} k q_{N-k} + \sum_{k=r+1}^{N} r q_{N-k}]/r. \tag{4.2}$$

For a particular job i if $q^{(i)}$ denotes the long run proportion of time that job i is at I/O processes, then

$$q^{(i)} = \sum_{n=1}^{N} \sum_{i \in \{i_1 \ldots i_n\}} Q_{i_1 \ldots i_n}. \tag{4.3}$$

Using a result due to Tomkó [13] we also have

$$q^{(i)} = 1/\lambda_i / \{1/\lambda_i + W_i + 1/\mu_i\} \tag{4.4}$$

where W_i is the mean time that job i waits not being serviced by a CPU. Thus

$$W_i = (1 - q^{(i)})/(\lambda_i q^{(i)}) - 1/\mu_i. \tag{4.5}$$

Of course with the queue discipline being considered the total waiting time may be made up of a number of such periods. The particular job may, at some stage, be in the selected set of those being serviced, and following a service completion or the arrival of another job may then not be in the selected set and will have to wait.

The mean response time of job i is given by

$$T_i = W_i + 1/\mu_i = (1 - q^{(i)})/(\lambda_i q^{(i)}) \tag{4.6}$$

so that the mean number of jobs that are calling for and receiving CPU service is given by

$$\bar{N} = \sum_{i=1}^{N} (1-q^{(i)}) = \sum_{i=1}^{N} \lambda_i T_i q^{(i)}. \tag{4.7}$$

Of course in the case of processor sharing as mentioned at the end of Section 2 there is no waiting time. However the mean response time as given by (4.6) is still appropriate for this discipline.

Acknowledgement

We are very grateful to Professor J. Tomkó who commented on an earlier version of this paper. His valuable and constructive criticism has, we believe, led to an improved presentation of this work.

SCHOOL OF MATHEMATICAL SCIENCES,
UNIVERSITY OF BRADFORD, U.K.

References

[1] H. ASHCROFT, The Productivity of Several Machines under the Care of One Operator, J. Roy. Stat. Soc. B, 12 (1) (1950), 145—151.
[2] B. AVI-ITZHAK and D. P. HEYMAN, Approximate Queueing Models for Multiprogramming Computer Systems, Opns. Res., 21 (1973), 1212—1230.
[3] F. BENSON and D. R. COX, The Productivity of Machines Requiring Attention at Random Intervals, J. Roy. Stat. Soc. B, 13 (1951), 65—82.
[4] B. D. BUNDAY and R. E. SCRATON, The G/M/r Machine Interference Model, Eur. J. Operation Res., 4 (1980), 399—402.
[5] J. W. COHEN, The Multiple Phase Service Network with Generalised Processor Sharing, *Acta Informatica*, 12 (1979), 245—284.
[6] D. P. GAVER, Probability Models for Multiprogramming Computer Systems, J. ACM, 3 (1967), 423—438.
[7] H. KAMEDA, A Finite-Source Queue with Different Customers, J. ACM, 29 (1982), 478—491.
[8] L. KLEINROCK, Queueing Systems, Vol. 2: Computer Applications, Wiley-Interscience, New York, 1976.
[9] L. G. PECK and R. N. HAZELWOOD, Finite Queueing Tables: ORSA Publications in Operations Research 2, Wiley, New York, 1958.
[10] P. SCHATTE, On the Finite Population GI/M/1 Queue and its Application to Multiprogrammed Computers, *Journal of Information Processing and Cybernetics*, 16 (1980), 433—441.
[11] J. SZTRIK, Probability Model for Non-Homogeneous Multiprogramming Computer System, *Acta Cybernetica*, 6 (1983), 93—101.
[12] J. SZTRIK, A Queueing Model for Multiprogrammed Computer Systems with Different I/O Times, *Acta Cybernetica*, 7 (1984), 127—135.
[13] J. TOMKÓ, Semi-Markov analysis of the inhomogeneous machine interference model: Lecture Notes in Control and Information Sciences, 84. System Modelling and Optimization. Proc. of the 12th IFIP Conf. Hungary, 1985, 992—1001.

(Received March 27, 1987, revised Nov. 3, 1987)

Nonlinear Parameter Estimation by Global Optimization — Efficiency and Reliability

T. CSENDES

1. Introduction

Our original task [8] was to solve parameter estimation problems having very complex nonlinear objective functions with a relatively small number of parameters. Their evaluation was quite expensive: about 100 times as much CPU time is needed as to the standard test functions. The objective functions usually turned out to have a large number of local minima in the region of interest. Although we can compute these functions, at times we do not even know their explicit form. Thus, determination of the exact, analytical derivatives is impossible in such cases, and we are forced to use non-derivative techniques.

The literature on global optimization [5] suggested that the method of Boender et al. [2] was the most promising for our purposes. Although a later version of this algorithm [11] seemed to be more efficient, we did not implement this modification because it was less reliable.

In this paper we discuss the relationship between the nonlinear least squares problem and global optimization, and we deal with the efficiency and reliablity of the above global optimization method using a quasi-Newton procedure and a random walk direct search technique.

2. Nonlinear parameter estimation and global optimization

The nonlinear parameter estimation problem is usually given as

$$\min_{\underline{x}} F(\underline{x}) \tag{1}$$

where $F(\underline{x})$: $\mathbf{R}^n \to \mathbf{R}$,

$$F(\underline{x}) = \left(\sum_{i=1}^{m} (\bar{f}_i - f_i(\underline{x}))^2\right)^{1/2}$$

$\bar{f}_i \in \mathbf{R}$; $f_i(\underline{x})$: $\mathbf{R}^n \to \mathbf{R}$, $i=1, 2, \ldots, m$; $m>0$ integer; $\underline{x} \in S \subseteq \mathbf{R}^n$, where the region of interest S is a compact set. S is in most cases a hypercube $a_i \leqq x_i \leqq b_i$; $a_i, b_i \in \mathbf{R}$ $i=1, 2, \ldots, n$. Thus the objective function $F(\underline{x})$ is of least squares type.

In solving (1), it is often supposed that $F(\underline{x})$ is unimodal (it has only one local minimum) or that a suitable starting point is at hand for the iterative solving algorithm [6]. Since we have found many practical nonlinear parameter estimation problems whose objective functions were not unimodal, we examine the relationship between the nonlinear least squares problem (1) and the global optimization problem:

Consider a compact set S in $\mathbf{R}^n$ and a not necessarily unimodal function $G(\underline{x})$: $\mathbf{R}^n \to \mathbf{R}$. The problem is to find a global minimizer $\underline{x}^* \in S$ such that $G(\underline{x}) \geqq G(\underline{x}^*)$ for all $\underline{x} \in S$.

S is usually given by simple bounds on the parameters of $G(\underline{x})$:

$$a_i \leqq x_i \leqq b_i, \quad a_i, b_i \in \mathbf{R}, \quad i = 1, 2, \ldots, n$$

We found that the structure of $F(\underline{x})$ guarantees only the non-negativity of $F(\underline{x})$. More exactly:

Proposition. *For every non-negative real function $G(\underline{x})$: $\mathbf{R}^n \to \mathbf{R}$, positive integer m and real numbers $\bar{f}_i \in \mathbf{R}$, $i=1, 2, \ldots, m$, there are real functions $f_i(\underline{x})$, $i=1, 2, \ldots, m$ such that*

$$F(\underline{x}) = \left(\sum_{i=1}^{m} (\bar{f}_i - f_i(\underline{x}))^2\right)^{1/2}$$

and $G(\underline{x}) = F(\underline{x})$ for every $\underline{x} \in \mathbf{R}^n$.

For example, let $g_i(\underline{x})$, $i=1, 2, \ldots, m$ be real, non-negative functions such that

$$G(\underline{x})^2 = \sum_{i=1}^{m} g_i(\underline{x})$$

There exist such functions $g_i(\underline{x})$, since $g_i(\underline{x}) = G(\underline{x})^2/m$ is suitable, for instance. Then, let

$$f_i(\underline{x}) = g_i(\underline{x})^{1/2} + \bar{f}_i.$$

Note that the functions $g_i(\underline{x})$ can be almost freely chosen, and in this way we can ensure further desirable properties of the functions $f_i(\underline{x})$. For example, when $G(\underline{x})$ is continuous, then all $f_i(\underline{x})$ can be continuous, too. On the other hand, for all sets of functions $f_i(\underline{x})$, $i=1, 2, \ldots, m$ there obviously exists a real function $f(i, \underline{x})$: $\mathbf{R}^{n+1} \to \mathbf{R}$, so that $f_i(\underline{x}) = f(i, \underline{x})$ for all $i=1, 2, \ldots, m$; and $f(i, \underline{x})$ is even continuous in the variable i.

According to the Proposition, the objective function of a nonlinear parameter estimation problem can be any non-negative real function. Thus, a nonlinear parameter estimation problem can have an arbitrary large number (or even a continuum) of local minima. The structure of $F(\underline{x})$, i.e. the least squares form, results only in the non-negativity of $F(\underline{x})$, and not in any further regularity.

Since the global minimum of a well-posed global optimization problem is finite (e.g. $G(\underline{x}^*) \in \mathbf{R}$), every such problem can be transformed with $G'(\underline{x}) = G(\underline{x}) - G(\underline{x}^*)$ and $S' = S$ to a problem having a non-negative objective function and the same structure of local minima. Thus, loosely speaking, every global optimization problem can be written in the form of a nonlinear parameter estimation problem with any m, and $\bar{f}_i$ values fixed in advance. This confirms the use of a global optimization algorithm to solve problems such as (1).

3. Implementations

We discuss here an algorithm to solve the global optimization problem defined in the previous section. In most cases, the result of a global optimization algorithm is only an approximation of the global optimum, though the precision of the modern sophisticated nonlinear optimization methods approaches that of the given computer.

The global optimization method of Boender et al. [2] has been implemented in two versions. These have the same structure, the only difference between them being the local search procedure (an algorithm to find a local minimizer) used: a quasi-Newton procedure with the DFP update formula [6] and a random walk direct search method UNIRANDI [9, 12]. In the following these algorithms are denoted by A and B, respectively. Both are derivative-free, i.e. they do not use the partial derivatives of the objective functions. Evaluation of the latter would be difficult or even impossible in the case of our original problem [8]. UNIRANDI proved to be robust but inefficient, whereas the quasi-Newton method was rather sensitive to the initial points but more accurate [3]. The global optimization method and UNIRANDI were implemented by using solely [2] and [12].

The global optimization algorithm discussed in this paper can be described concisely as follows:

Step 1. Draw N points with uniform distribution in S, and add them to the current sample C. Construct the transformed sample T by taking the γ percent of the points in C with the lowest function values.

Step 2. Apply the clustering procedure to T. If all points of T can be assigned to a cluster, go to Step 4.

Step 3. Apply the local search procedure to the points in T not yet clustered. Repeat Step 3 until every point has been assigned to a cluster. If a new local minimizer has been found, go to Step 1.

Step 4. Determine the smallest local minimum value found, and stop.

The local search procedure mentioned here is either the quasi-Newton method or UNIRANDI. We chose the single linkage clustering procedure as being the more promising of the two discussed in [2]. The aim of this procedure is to recognize those sample points starting from which the local search would possibly result in an already found local minimizer. Clusters are grown around seed points (local minimizers or such points of the local search procedure from which an already known local minimum was reached). A distance $d(\underline{x}, \underline{x}')$ is defined [2] for the clustering between two points $\underline{x}$ and $\underline{x}'$ in the neighbourhood of a local minimizer $\underline{x}^*$ by

$$d(\underline{x}, \underline{x}') = ((\underline{x}-\underline{x}')^T H(\underline{x}^*)(\underline{x}-\underline{x}'))^{1/2}.$$

The quasi-Newton method of algorithm A gives a good approximation to the Hessian $H(\underline{x}^*)$ of the objective function. In the case of UNIRANDI the identity matrix replaces $H(\underline{x}^*)$ (cf. [2]). A new point x is added to a cluster if there is a point x' in this cluster for which

$$d(\underline{x}, \underline{x}') \leqq \left[\frac{\Gamma\left(1+\frac{1}{2}n\right)|H(\underline{x}^*)|^{1/2} m(S)}{\pi^{n/2}} (1-\alpha^{1/(N'-1)})\right]^{1/n}$$

where $|H(\underline{x}^*)|$ denotes the determinant of $H(\underline{x}^*)$, $m(S)$ is a measure of the set S, N' is the total number of sampled points, and $0<\alpha<1$ is a parameter of the clustering procedure [2].

The two most important changes in the original algorithm are as follows:

1. We do not use a steepest descent step to transform the current sample. Its efficiency was examined in the early phase of the implementation, and it turned out to be omittable.
2. The parameters of the objective function are scaled [6] by the global optimization subroutine with the transformations

$$x_i' = \frac{2x_i - a_i - b_i}{b_i - a_i} \quad i = 1, 2, \ldots, n.$$

This can be done, of course, without using the explicit form of the objective function. The scaling does not have much effect on the efficiency of algorithms A and B in the case of the test functions. On the other hand, it is indispensable for the solution of practical problems.

The result of the implementation was a FORTRAN subroutine of just over 400 program lines, occupying 44 kilobytes of core space (without the local search routines). It serves to solve global optimization problems of up to 15 parameters. The program documents its progress, and when the problem is solved it makes a list of local minima with increasing function values.

4. Efficiency

The numerical tests were carried out on a ROBOTRON R55M computer. The programs were coded to use single precision arithmetic (with 7.2 decimal digits). The standard time unit (1000 evaluations of the S5 function at $\underline{x}^T=(4.0, 4.0, 4.0, 4.0)^T$) was measured ten times. The average of these was 2.00 seconds with a standard deviation of 0.15. We used the usual test functions whose detailed description can be found in [5], [4] and [7]. With these functions, mostly the efficiency of a global optimization algorithm can be measured. Wherever possible, the results from the original papers are included in our tables. These data differ slightly from those in [5] and [2]. Algorithms A and B were applied to each test function ten times. The parameters of the procedures were chosen so that they were able to find the global minimum each time.

We found that the computational effort (CPU time and number of function evaluations) was proportional to the required precision of the estimation of local minima. Thus, when different global optimization methods are compared, their accuracy should also be taken into account. First of all, the exact global minimum values should be determined. Table 1 gives the accurate global minimum and global minimizer values for every test function. These data are in good agreement with the results of Price [10]. It should be mentioned that slightly different numbers can be obtained with another computer precision. Certain global minimizer values in Table 1 are given by four or five decimal digits only, since for these test functions the same global minimum value can be achieved with somewhat different minimizers.

We subsequently determined the precision of the results obtained with the global

Table 1: *Global minimum and global minimizers of the test functions*

Test function	$F(\underline{x}^*)$	x_1^*	x_2^*	x_3^*	x_4^*	x_5^*	x_6^*
S5	−10.153206	3.99995	4.00014	4.00011	4.00016		
S7	−10.402947	4.00061	4.00072	3.99945	3.99958		
S10	−10.536416	4.00075	4.00061	3.99967	3.99948		
H3	−3.8627815	0.1146	0.5557	0.8525			
H6	−3.3223667	0.201536	0.149909	0.476906	0.275239	0.311593	0.657353
GP	2.9996490	0.000068	−1.0001				
RCOS	0.39788723	−3.1416	12.275				
	0.39788723	3.1416	2.2750				
	0.39788723	9.425	2.4750				
SHCB	−1.0316286	0.0899	−0.7126				
	−1.0316286	−0.0899	0.7126				
RB	0.0	1.0	1.0				

optimization methods cited in [2] wherever this was possible. The number of significant digits was defined by

$$-\log\frac{F(\underline{x}')-F(\underline{x}^*)}{F(\underline{x}^*)} \tag{2}$$

where $\underline{x}^*$ is a global minimizer of the given test function $F(\underline{x})$, and $\underline{x}'$ is its estimate. In the particular case of the Rosenbrock function (RB) where the global minimum is zero, the following expression was used for this:

$$-\log F(\underline{x}').$$

The numbers of significant digits are listed in Table 2 for every test function. We tried to tune the procedures A and B so that they achieve a similar accuracy (2) on the various test functions. The reliability and the accuracy of our method can be tuned almost independently.

The numbers of function evaluations required by the global optimization methods to solve test functions are given in Table 3. Since those local minimization

Table 2: *Number of significant digits in the global minimum*

Method	Test function								
	S5	S7	S10	H3	H6	GP	RCOS	SHCB	RB
Branin	—	—	—	—	—	—	—	—	—
Törn*	3.0	2.9	3.2	4.3	2.6	3.9	4.5	—	—
Price*	6.2	5.3	5.8	6.4	6.0	3.9	6.3	—	—
De Biase	2.9	3.4	2.0	4.7	4.7	4.8	—	6.4	—
Boender	—	—	—	—	—	—	—	—	—
A*	7.0	6.8	6.7	6.8	6.9	4.3	7.2	7.1	10.1
B*	4.9	4.0	5.2	4.3	4.3	4.4	4.1	4.5	4.7

* These methods do not use the partial derivatives of the objective function.

Table 3: *Number of function evaluations*

Method	Test function								
	S5	S7	S10	H3	H6	GP	RCOS	SHCB	RB
Branin	5500	5020	4860	—	—	—	—	—	—
Törn*	3649	3606	3874	2584	3447	2499	1558	—	—
Price*	3800	4900	4400	2400	7600	2500	1800	—	—
De Biase	620	788	1160	732	807	378	597	717	—
Boender	567	624	755	235	462	398	235	—	—
A*	990	1767	2396	216	1446	436	330	233	410
B*	1083	1974	2689	697	2610	386	464	267	1524

* These methods do not use the partial derivatives of the objective function.

procedures that are not allowed to use the partial derivatives of the objective function are usually less efficient than the others, the efficiencies of algorithms A and B should be compared only with those of the similar non-derivative methods. The methods known to be non-derivative are marked by asterisks in Tables 2—4. The numbers in these Tables are results of a single sample run for each of the first four methods, the average of four independent runs for the method of Boender et al [2], and the averages of ten runs for algorithms A and B. Table 3 indicates that the procedure of Boender et al. works best of all, and the non-derivative methods of Törn [12] and Price [10] are less efficient than A and B.

Table 4 contains the numbers of standard time units required. As concerns these data, algorithms A and B seem to be definitely quicker than the other non-derivative ones, and procedure A is about as rapid as that of Boender et al. [2]. From the user's point of view Table 3 is more important, since in practical cases the evaluation of the objective function is more expensive than that of the standard test functions. Therefore, Table 4 is informative as to the overhead costs.

To summerize our numerical experience, we can state that these two non-derivative versions of the global optimization method of Boender et al. work definitely better than the other non-derivative procedures. The efficiency of implementation A

Table 4: *Numbers of standard time units*

Method	Test function								
	S5	S7	S10	H3	H6	GP	RCOS	SHCB	RB
Branin	9.0	8.5	9.5	—	—	—	—	—	—
Törn*	10.0	12.4	14.4	8.0	15.6	4.1	3.7	—	—
Price*	13.9	20.0	19.7	7.5	47.5	2.8	4.4	—	—
De Biase	26.1	23.0	33.7	17.6	23.1	16.8	15.2	23.2	—
Boender	3.5	4.5	7.0	1.7	4.3	1.5	1.0	—	—
A*	3.0	4.9	7.0	1.2	4.2	1.3	1.4	1.2	1.0
B*	3.5	6.0	8.8	1.9	14.2	1.5	1.6	1.3	1.5

* These methods do not use the partial derivatives of the objective function.

approaches that of the original one. The discussed global optimization method with a non-derivative quasi-Newton procedure can be highly recommended for the solution of smooth global optimization problems when calculation of the partial derivatives is inconvenient or impossible. The same global optimization method, together with the direct search method UNIRANDI, can be an efficient tool for locating the global minimum of non-smooth or non-differentiable objective functions.

5. Reliability

Almost all global optimization methods use only local information, i.e. the values of the objective function and its first and second derivatives at certain points. It is easy to show that, for the solution of this problem in general, there is no algorithm that uses only such local information at a finite number of points. For this reason, the research efforts on global optimization are concetrated mainly on evaluating increasingly reliable and efficient heuristics.

The size of the region of attraction [1] of the global minimum (the points of continuous curves in $\mathbf{R}^n$ that end in a global minimizer, and along which the objective function decreases strictly monotonously) characterizes the difficulty of a given problem. From this point of view, the most frequently used test problems [5] are rather easy to solve, and mostly the efficiency of an algorithm can be tested with them.

A new global optimization test problem is proposed below for comparing algorithms in terms of reliability and for testing the degree of difficulty of global optimization problems that can be solved with them.

The suggested n-dimensional test function is very simple:

$$F(\underline{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{3}$$

where for every $i = 1, 2, \ldots, n$:

$$f_i(x_i) = x_i^6 \left(\sin(1/x_i) + 2\right)$$

if $x_i \neq 0$, and

$$f_i(0) = 0.$$

If $x_i \neq 0$, the gradient and Hessian of $F(\underline{x})$, respectively, are

$$g_i(\underline{x}) = 6x_i^5 \left(\sin(1/x_i) + 2\right) - x_i^4 \cos(1/x_i) \tag{4}$$

and

$$H_{i,j}(\underline{x}) = 0 \quad (i \neq j)$$

$$H_{i,i}(\underline{x}) = 30x_i^4 \left(\sin(1/x_i) + 2\right) - 10x_i^3 \cos(1/x_i) - x_i^2 \sin(1/x_i) \tag{5}$$

$i, j = 1, 2, \ldots, n$. Otherwise, $g_i(x)$ and $H_{i,j}(\underline{x})$ are zeros, $i, j = 1, 2, \ldots, n$. The gradient and the Hessian are continuous everywhere in $\mathbf{R}^n$. Since

$$\sum_{i=1}^{n} x_i^6 \leqq F(\underline{x}) \leqq 3 \sum_{i=1}^{n} x_i^6 \tag{6}$$

the global minimum of $F(\underline{x})$ on $\mathbf{R}^n$ is zero, and this value is reached only in the origin.

Theorem. *The function $F(\underline{x})$ has a countable infinity of local minima and maxima. All these extrema are in the hypercube*

$$-1 \leqq x_i \leqq 1 \quad i = 1, 2, \ldots, n. \tag{7}$$

Proof. First consider the case when $n=1$. Supposing that the first derivative is equal to zero and $x \neq 0$, it holds that

$$6x(\sin(1/x)+2) = \cos(1/x). \tag{8}$$

The right side of this equation varies from -1 to 1, while the left side is between $6x$ and $18x$. Hence, the first derivative can be zero only in the interval $(-1/6, 1/6)$. The right side of equation (8) takes the values -1 and 1 in each interval of

$$[(1/2k\pi), 1/(2(k+1)\pi)) \quad k = \pm 3, \pm 4, \ldots \tag{9}$$

whereas in the same interval

$$-1 < 6x(\sin(1/x)+2) < 1. \tag{10}$$

This proves that the function $F(x)$ has at least one local minimum and one local maximum in every interval of (9), since the first derivative is a continuous function. These extrema are diverse, because they are all inside the intervals. Thus, there is at least a countable infinity of local minima and maxima in (7).

It can easily be seen that the first and the second derivatives of $F(x)$ can not be zeros in the same place. Since the second derivative is continuous, each local extremum is associated with an open interval of **R**, in which it is the only local extremum. Consequently, there cannot be a continuum of local extrema of $F(x)$ in **R**.

For any positive integer n the same proof holds, by using the fact that $F(\underline{x})$ is separable. □

Thus, the unconstrained problem has the same set of local minima as the problem with the bounds (7). The global minimizer is non-isolated, in the sense that it is an accumulation point of local minimizers [1] (and it is the only one). The region of attraction of the global minimum is obviously of zero measure. The most important property of $F(\underline{x})$ is that the smaller the local minimum, the smaller the measure of the region of attraction relating to this local minimum. This feature can be used to asses the degree of difficulty of global optimization problems that can be solved via the given method.

The local minimizers of the one-dimensional version of the test function can be ordered according to the magnitude of the function value. The serial number N_x of the local minimizer x can be calculated using the equation

$$N_x = 2\lfloor |1/x|/2\pi \rfloor - 1 + (\operatorname{sgn}(x) - 1)/2 \tag{11}$$

where $\lfloor \cdot \rfloor$ denotes the largest integer not greater than the argument, and *sgn* stands for the signum function. In the one-dimensional case the size of the region of attraction A_x of local minimizer x can be well estimated by using equation (8), provided that the absolute value of x is small. The left side of this equation is then close to zero, and A_x is approximately equal to the distance between the two zeros of the right side of equation (8) that are adjacent to x:

$$A_x \approx \frac{2}{\frac{1}{x_2} - \pi^2}. \tag{12}$$

The numerical form of $F(x)$ obviously differs from the analytical one, especially near the origin. Thus, it is important to code this test function very carefully. Our version was written in FORTRAN and run on the mainframe R55M, by using single precision arithmetic.

The proposed test function can be computed quickly: in the one-dimensional case 1000 evaluations of $F(x)$ need 0.306 ± 0.006 (SD) standard time units [5]. When $n=4$, the corresponding figure is 0.829 ± 0.001 (SD). Accordingly, the computation of even the four-dimensional version requires somewhat less computational effort than that of the S5 function [5]. The numerical form of $F(x)$ is zero in the hypercube $x_i \in (-1.0\ 10^{-13}, 1.0\ 10^{-13})$, $i=1, 2, \ldots, n$. In spite of this, there are more than one million local minima whose regions of attraction contain at least 100 points that can be represented by using single precision.

The algorithm A was tested by running it independently ten times on the one- and four-dimensional versions of this test problem with the bounds (7). The parameters of the algorithm were set so that the estimate of the global minimum was as close to zero as possible, and they were different from those used in the previous section. From the point of view of this reliability test the type of the local search procedure is indifferent.

In the one-dimensional case, the smallest minimum found was $0.523449\ 10^{-52}$ in $0.193281\ 10^{-8}$. This was the 164,687,623rd local minimizer in the sequence discussed above, and the size of its region of attraction A_x was $0.23472\ 10^{-16}$ according to (12). The worst estimate of the global minimum was $0.319144\ 10^{-23}$ in $-0.119009\ 10^{-3}$; this was the 2,673rd local minimizer, with $A_x = 0.88989\ 10^{-7}$. The average run consumed 33.5 standard time units and 22,137 function evaluations. In the four-dimensional case, the best and the worst estimates of the global minimum were $0.272099\ 10^{-8}$ and $0.598347\ 10^{-6}$, respectively. The average run consumed 46.1 standard time units and 22,020 function evaluations.

In conclusion, the results of this reliability test have shown that the studied global optimization method can be tuned to solve most practical problems with satisfactory reliability.

Abstract

In this paper we first show that the objective function of a least squares type nonlinear parameter estimation problem can be any nonnegative real function, and therefore this class of problems corresponds to global optimization. Two non-derivative implementations of a global optimization method are presented, with nine standard test functions applied to measure their efficiency. A new nonlinear test problem is then presented for testing the reliability of global optimization algorithms. This test function has a countable infinity of local minima and only one global minimizer. The region of attraction of the global minimum is of zero measure. The results of efficiency and reliability tests are given.

Key words. Global optimization, nonlinear parameter estimation, sum of squares, least squares, test problem.

T. CSENDES
KALMÁR LABORATORY OF CYBERNETICS
JÓZSEF ATTILA UNIVERSITY
ÁRPÁD TÉR 2.
SZEGED
HUNGARY

References

[1] F. Archetti, G. P. Szegő: Global optimization algorithms, in: Nonlinear Optimization — Theory and Algorithms (Dixon, L. C. W., Spedicato, E., Szegő, G. P., eds., Birkhäuser, Boston, 1980), 429—469.
[2] C. G. E. Boender, A. H. G. Rinnooy Kan, G. T. Timmer, L. Stougie: A stochastic method for global optimization, *Math. Programming*, 22 (1982), 125—140.
[3] T. Csendes, B. Daróczy, Z. Hantos: Nonlinear parameter estimation by global optimization: comparison of local search methods in respiratory system modelling, in: System Modelling and Optimization (Thoma, M., Wyner, A., eds., Springer-Verlag, Berlin, 1986), 188—192.
[4] L. De Biase, F. Frontini: A stochastic method for global optimization: its structure and numerical performance, in: Towards Global Optimisation 2. (Dixon, L. C. W., Szegő, G. P., eds., North-Holland, Amsterdam, 1978), 85—102.
[5] L. C. W. Dixon, G. P. Szegő: The global optimisation problem: an introduction, in: Towards Global Optimisation 2. (Dixon, L. C. W., Szegő, G. P., eds., North-Holland, Amsterdam, 1978), 1—15.
[6] P. E. Gill, W. Murray, M. H. Wright: Practical Optimization (Academic Press, London, 1981).
[7] J. Gomulka: Deterministic vs probabilistic approaches to global optimisation, in: Towards Global Optimisation 2. (Dixon, L. C. W., Szegő, G. P., eds., North-Holland, Amsterdam, 1978), 19—29.
[8] Z. Hantos, B. Daróczy, B. Suki, G. Galgóczy, T. Csendes: Forced oscillatory impedance of the respiratory system at low frequencies. J. Appl. Physiol. 60 (1986), 123—132.
[9] T. Järvi: A random search optimizer with an application to a maxmin problem, *Publications of the Institute for Applied Mathematics*, University of Turku, No. 3, 1973.
[10] W. L. Price: A controlled random search procedure for global optimisation, in: Towards Global Optimisation 2. (Dixon, L. C. W., Szegő, G. P., eds., North-Holland, Amsterdam, 1978), 71—84.
[11] G. T. Timmer: Global optimization: a stochastic approach, (Ph. D. Thesis, *Erasmus University*, Rotterdam, 1984).
[12] A. A. Törn: A search-clustering approach to global optimization, in: Towards Global Optimisation 2. (Dixon, L. C. W., Szegő, G. P., eds., North-Holland, Amsterdam, 1978), 49—62.

(Received Dec. 30, 1987.)

Bibliographie

B. D. Craven: Fractional Programming, Heldermann Verlag, Berlin, 1988. (Sigma Series in Applied Mathematics; Vol. 4)

From preface of the book: A linear program optimizes a linear function of several variables, subject to linear constraints. Linear programming models have been extensively applied in management, industry, and economics. But, quite often, the function to be optimized is, instead, a ratio of two linear functions. The term fractional programming describes the larger class of optimization problems, where the objective function to be optimized is a ratio. The ratio form gives such problems additional properties, which not all nonlinear programming problems have.

This book describes fractional programming from the standpoints of applications (potential and actual), the mathematical theory (including duality and analysis of sensitivity to perturbations), and algorithms by which optima of fractional programs may be computed.

Chapter 1 surveys many applications proposed for fractional programs to problems of management, scheduling, and finance. Both linear fractional programs (with a ratio of linear functions), and nonlinear fractional programs (with a ratio of nonlinear functions), are considered.

Chapter 2 presents the theory of linear fractional programming, including duality, and equivalent programs.

Chapter 3 discusses nonlinear fractional programming, especially maximizing the ratio of a concave to a convex function.

Chapter 4 deals with various aspects of duality, sensitivity to perturbations, and recent improvements (invex functions, quasiduality) which extend results to more functions.

Chapter 5 surveys various algorithms, which can compute an optimum of a fractional program.

Chapter 6 outlines some further recent developments, including optimization with several objective functions.

Each chapter, after the first, includes a set of exercises. Each chapter includes a bibliography of references cited, from the very large literature.

The book is clearly written and may be recommended as a textbook for students in a lecture course on fractional programming.

J. Csirik

STACS 88, 5th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 1988. Proceedings, Springer Lecture Notes in Computer Science Vol. 294. R. Cori, M. Wirsing (Eds.), IX, 404 pages. 1988.

The volume contains an invited paper and 34 contributed papers presented at the 5th STACS conference. STACS is a regular conference on theoretical computer science, held each year, alternately in France and West Germany.

The papers are classified according to their topic. In the algorithmic, complexity-theoretic direction the sections are Algorithms, Complexity, Distributed Algorithms and Geometrical Algorithms. In the algebraic, formal language-theoretic direction the sections are Formal Languages, Rewriting Systems and Abstract Data Types, Graph Grammars, Trace Languages and Semantics of Parallelism. The volume also contains short descriptions of the software systems presented at the conference.

The contributed papers and systems demonstrations come from 13 countries, with the largest number of papers from France and West Germany. The research interests are indicated by the follow-

ing: 7 of the 9 papers with French authors are in the formal language sections and 8 of the 11 papers with West German authors are in the algorithms sections.

STACS is a well-established, high quality conference. The volume gives a good selection of current research topics in theoretical computer science and can be recommended to those who are interested in the state of the art of this field.

György Turán

M. Hofri: Probabilistic analysis of algorithms, (Texts and monographs in computer science), Springer-Verlag, 1987.

"Until quite recently, "analysis of algorithms" was nearly synonymous with determining the "Complexity class" of an algorithm. This has the objective, most often, of finding whether in all cases the running time (or storage requirements) of the algorithm operation is or is not bounded by some specified function of the size of a suitably devised representation of the problem. It usually boils down to the consideration of some extreme, especially crafted problem instances. The realization that there is more one could say to characterize the cost of using an algorithm is probably due to the influence of Knuth's series on "The Art of Computer Programming" which started out in 1968. There, clearly, the operation of algorithms was shown to be associated with probabilistic concepts and processes.

Random elements, and hence the call for stochastic analysis, may enter algorithms in essentially two ways. On the one hand, we find the so called "probabilistic algorithms", such that choose part of their actions on the basis of random elements, explicitly introduced into the algorithm specification (pseudo-random numbers, simulated coin flipping and the like). Numerous algorithms of this class where recently developed, some showing progress well beyond anything one has believed hitherto possible (primality testing algorithms provide a good example). On the other hand, we find the operation to deterministic algorithms on input data over wich some probability measure can be stipulated. While the sources of the randomness present a true dichotomy, the required analyses turn out typically to be of the same nature in both cases. Among the algorithms for which we provide detailed analyses, the reader will find examples of both varieties. While the analyses proper are similar, we show in Chapter 1 that the second type brings up methodological and conceptual problems that the first case need not entail. The difficulty there may be phrased as leading substance to the notation of two algorithms having the same complexity "on the average", or "in distribution". The problem may also be seen to reside in the attribution of a priori probability measures to the input instance space. At the time of writing there is no coherent accepted theory or even taxonomy for these vexing issues, comparable to standard complexity theory; we shall mostly skirt them, using reasonable — sometimes seemingly facile — assumptions, invoking naturalness as our guidline.

The probabilistic analysis of algorithms, as a discipline, draws on a fair number of branches of mathematics. Principally: probability theory (especially as applied to stochastic processes), graph theory, combinatorics, real and complex analysis, and occasionally algebra, number theory, computation theory, operational calculus and more. It was unreasonable to expect the students to have more than a cursory knowledge of most of the techniques we used, so much of the time was given over to introducing and exploring these methods as we went along. Arranging the text so it could be conveniently used both as a text and as a reference posed a problem which was solved by departing in the book version form the order of the class presentation to a large extent, collecting most of the methodological material in Chapter 2."

In next three Chapters some application areas are presented:

— in Chapter 3: Algorithms over Permutations (locating the largest Term in a Permutation; Representations of Permutations; Analysis of Sorting Algorithms),

— in Chapter 4: Algorithms for Communications Networks (The Efficiency of Multiple Connections; Collision Resolution Stack Algorithms),

— in Chapter 5: Bin Packing Algorithms (The Next—Fit Bin Packing Algorithm; The Next—Fit—Decreasing Bin Packing Algorithm).

This is a very well written book. It may be recommended for a large number of people from graduate students to researchers on given fields.

J. Csirik

A SZERKESZTŐ BIZOTTSÁG CÍME:

6720 SZEGED
SOMOGYI U. 7.

EDITORIAL OFFICE:

6720 SZEGED
SOMOGYI U. 7.
HUNGARY

Information for authors

Acta Cybernetica publishes only original papers in the field of computer sciences mainly in English, but also in French, German or Russian. Authors should submit two copies of manuscripts to the Editorial Board. The manuscript must be typed double-spaced on one side of the paper only. Footnotes should be avoided and the number of figures should be as small as possible. For the form of references, see one of the articles previously published in the journal. A list of special symbols used in the manuscript should be supplied by the authors.

A galley proof will be sent to the authors. The first-named author will receive 50 reprints free of charge.

INDEX — TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1988 február
Terjedelem: 7,84 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 6601 és az MSZ 5602—55 szabvány szerint
88-1052 — Szegedi Nyomda — Felelős vezető: Surányi Tibor igazgató